(Another) Intro To Git

How you can learn to love version control.

Redbrick
DCU's Networking Society

Our journey

Brief introduction to git.

I’m going to explain some of the fundamentals of git.

I’ll give you a demo.

Then it’s your turn.

Redbrick
DCU’s Networking Society

Pretext

Software tends to be written by more than one person.

At any given time, different people will be working on different features. Many changes are being made at once.

There needs to be a way to maintain the original source code while working on these features.

What is Git?

Git is a distributed version control system.

Designed to handle everything from small to large projects.

Created by Linus Torvalds in 2005.

Created to aid the development of the Linux kernel.

I DID NOT
CREATE GIT TO MAKE YOU SMILE
imgflip.com

What is version control?

A system which records files and their change over time so earlier working versions can be restored.

Users keep entire code and history on their local machines.

User can make changes to the codebase without Internet Access.

Except for pushing and pull (but we will get to that later).

Other Version Control Systems?

Yes there are.

We don’t talk about them.

How does git work?

Can be complicated at first. There’s a lot to git.

We are going to cover just the basics today.

Repositories (repos)

The structure which contains of all the files in your project.

Sites such as Github and Gitlab provide free hosting of repositories.

Redbrick has our own hosting service known as Gitea, found at https://git.redbrick.dcu.ie

Repositories (repos)

Repos consist of:

- all your additions to the project.
- the history of how the project has changed over time.

How does git track all this?

Snapshots

Git records what all your files look like at a given point in time.

You decide when to take a snapshot.

You decide what files are included in this snapshot.

How do we take this snapshot?

Commits

The acting of making a snapshot.

A commit can be seen as a change or addition to the repo.

Repos are made up of snapshots, i.e, commits.

What is a commit made up of?

Commits

Commits consist of:

- Info on how the files have changed.
- A reference to the previous commit, i.e the “parent commit”.
- Hash code, e.g - f449c4a98675d4bcba42600b3366df977c1bf44d

Quick Recap

What is version control?

What are repositories?

What are repositories made up of?

Copying Repos

Repos can live on local machine or remote server.

Cloning = copying from the remote server to your local machine

Cloning allows teams to work together

Time to use git

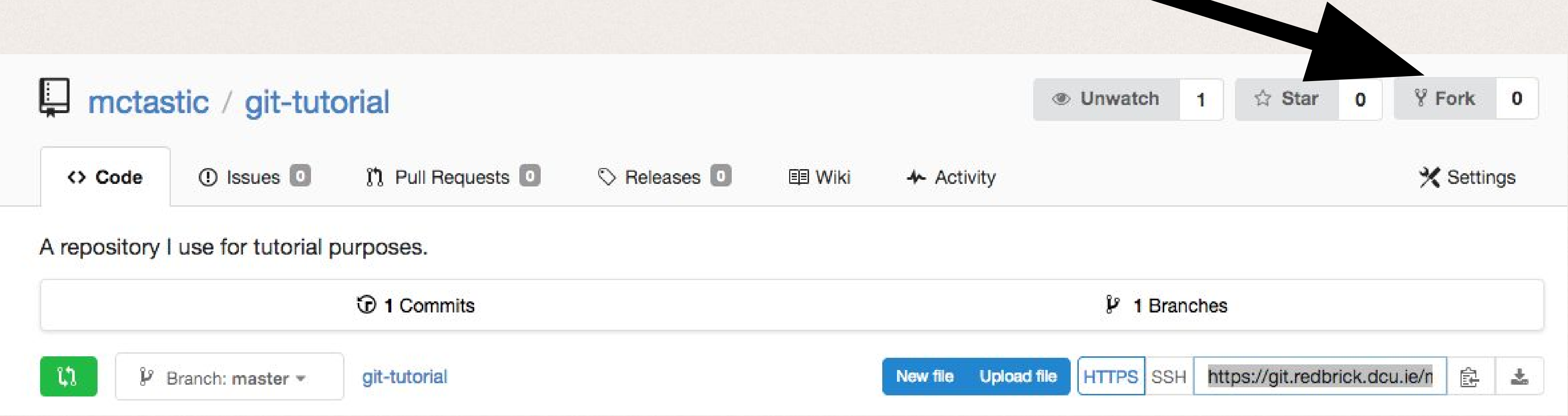
Login on git.redbrick.dcu.ie with your Redbrick username & password.

Go to https://git.redbrick.dcu.ie/mctastic/git-tutorial

Time to use git

Open up your terminal.

Fork this repo



A repository I use for tutorial purposes.

https://git.redbrick.dcu.ie/n

Redbrick
DCU's Networking Society

Time to use git

Copy the HTTPS link on your fork and in your terminal type:

```
git clone
```

Followed by the copied link

You now have a local copy of the remote repository.

Redbrick
DCU's Networking Society

Lets run it

We have an extra file

Some files are generated when you compile a program.

Typing:

```
git status
```

Shows us which files have been changed/created.

We have an extra file

Now there’s a new file called Factorial.class

We want to ignore this.

We make a new file called .gitignore, and inside this file we put the line:

```
* .class
```

This ignores all files ending in .class

Pulling

When remote repo commits > local machine commits.

You can’t add anything new until you pull the remote changes to your local repo.

Pulls remote changes to local repo.

There’s a new commit

Make an empty factorial.py in your remote and then in your terminal type:

```
git pull
```

This adds the newly added factorial.py script from your remote fork to your local repo.

Pushing

Adding your local changes to the remote repo is called pushing.

Takes place after adding and commit your changes.

Push local changes to remote repo.

Lets run it

Write factorial.py

Now it’s your turn. Write a working version of factorial.py in python3.
When you’ve written a working version of factorial.py, do the following:

```
 git add factorial.py
 git commit -m “created factorial.py”
git push origin master
```

Redbrick
DCU’s Networking Society

What else can git do?

Too much to talk about in this length of time.

Some things we didn't cover today:

branching

merging

rebasing

Questions?

Redbrick
DCU's Networking Society