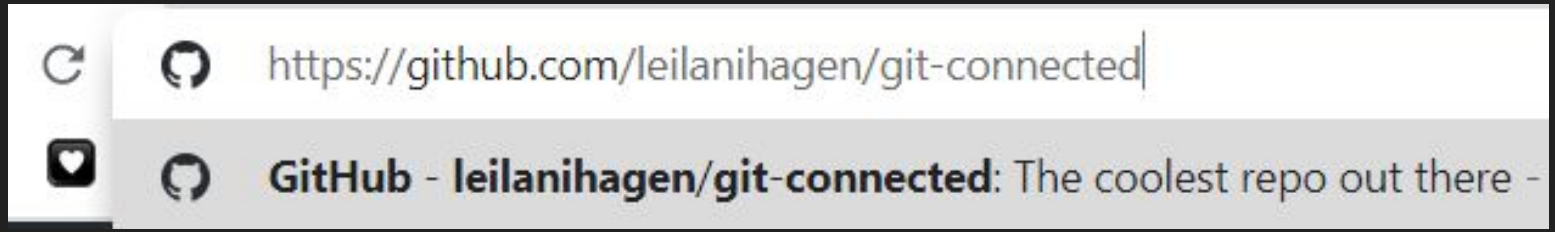
Git-Connected

Get our slides!

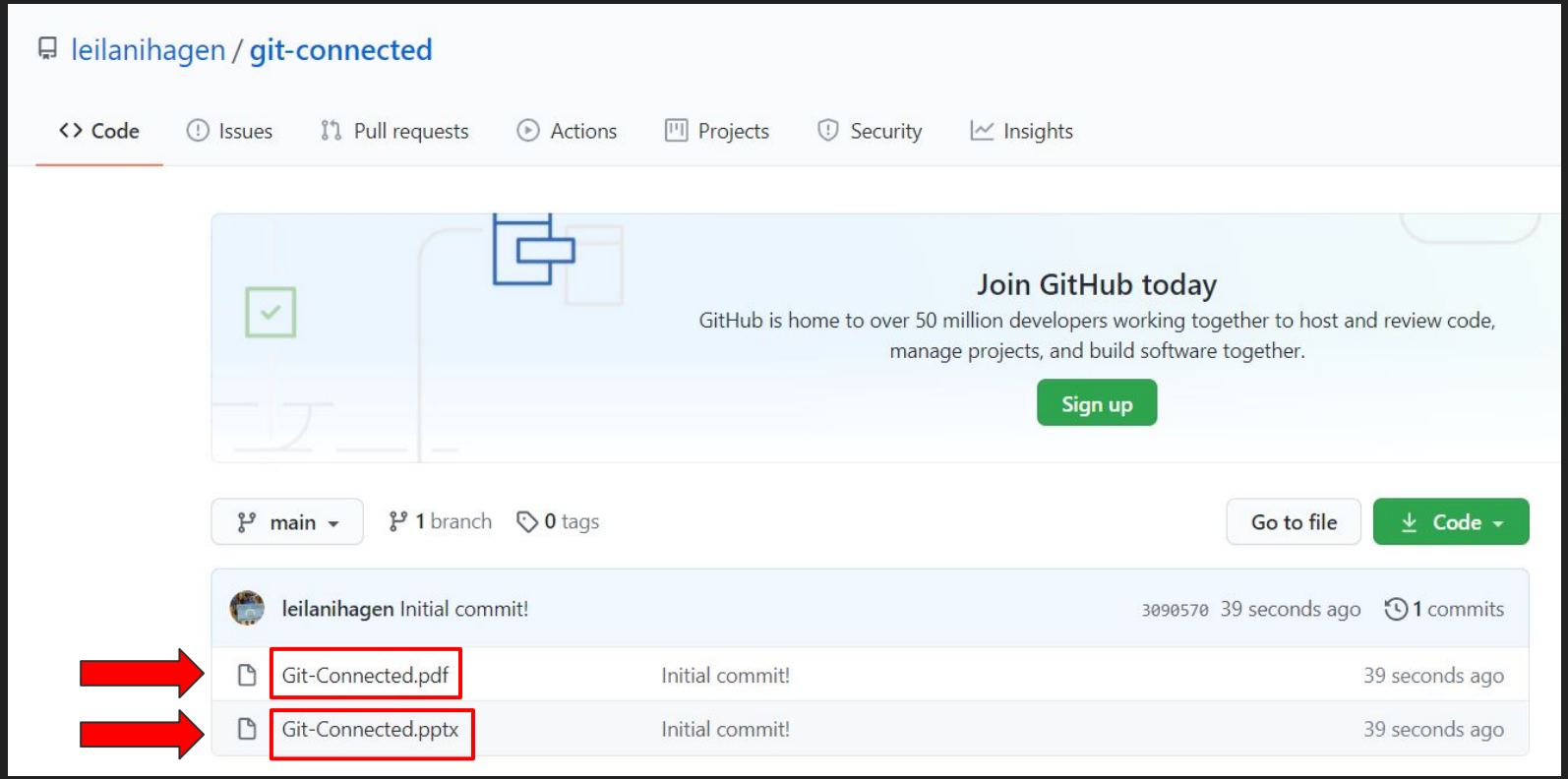
1. 

Go to https://github.com/leilanihagen/git-connected

(No GitHub account necessary)

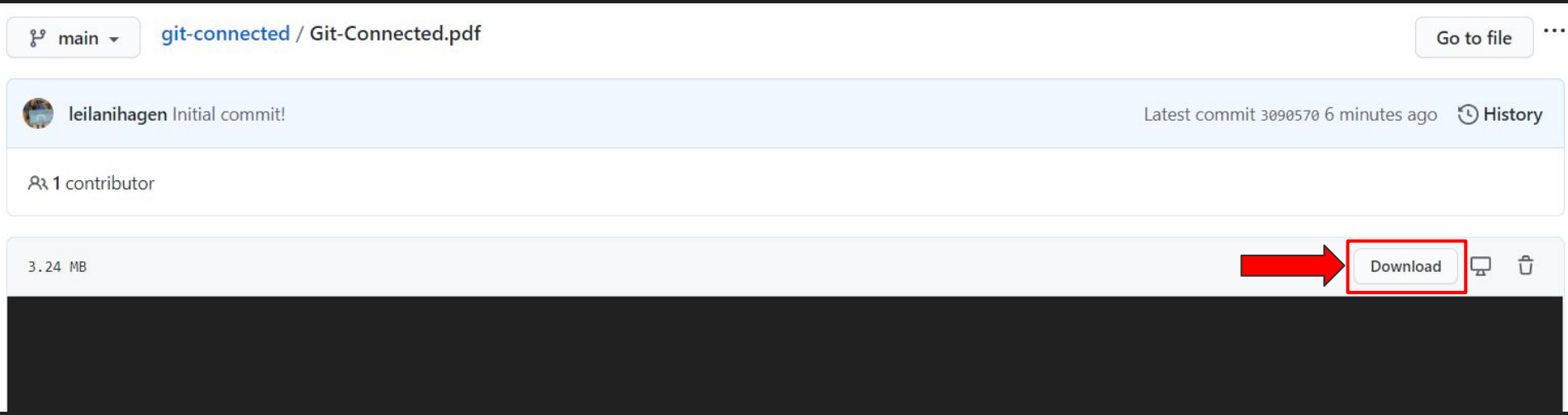
Get our slides!

2.

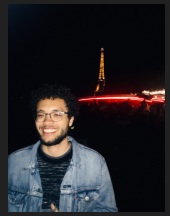


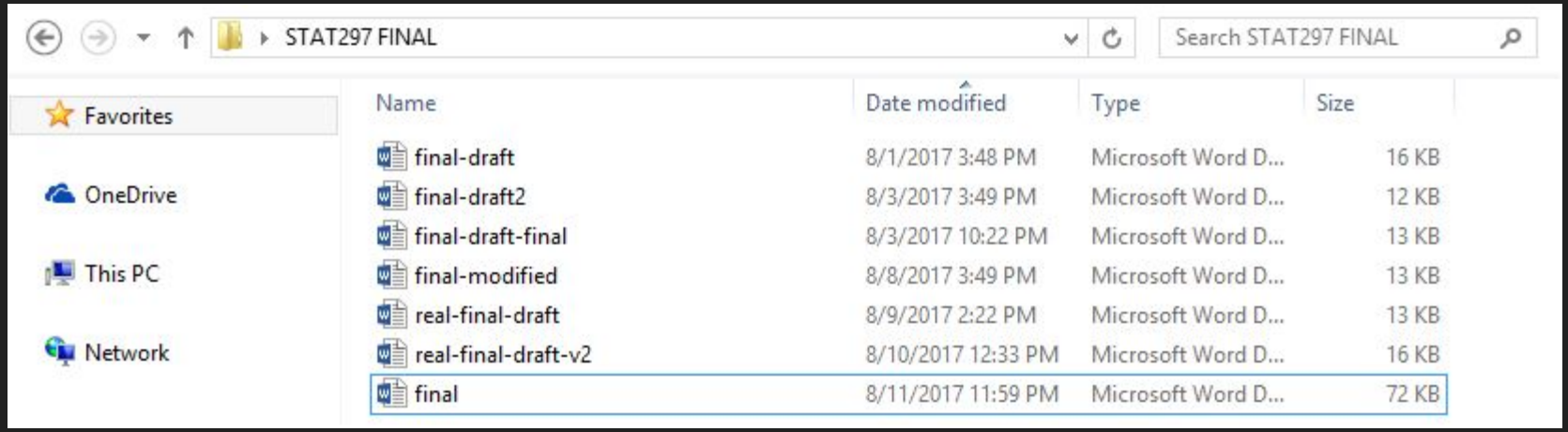
Get our slides!

3.



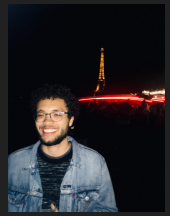
File management



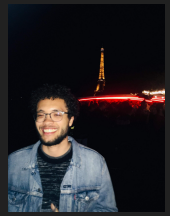


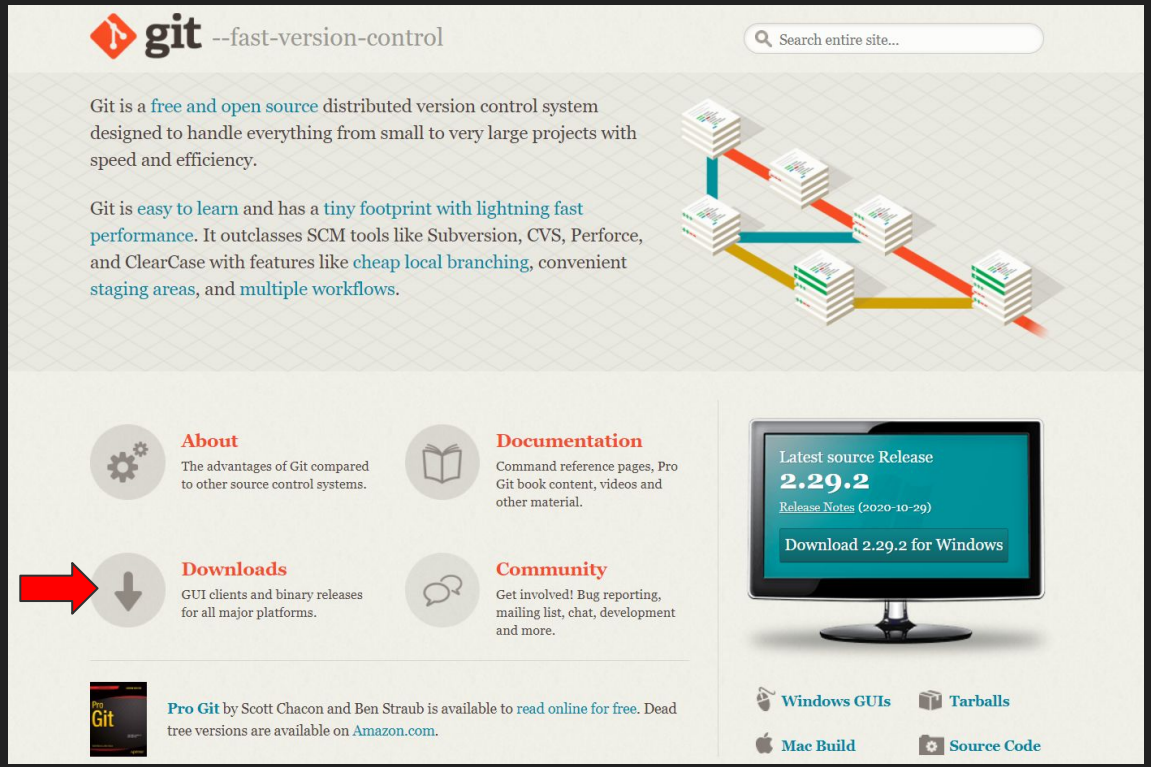
Git

- Fast
- Secure
- Offline
- Distributed



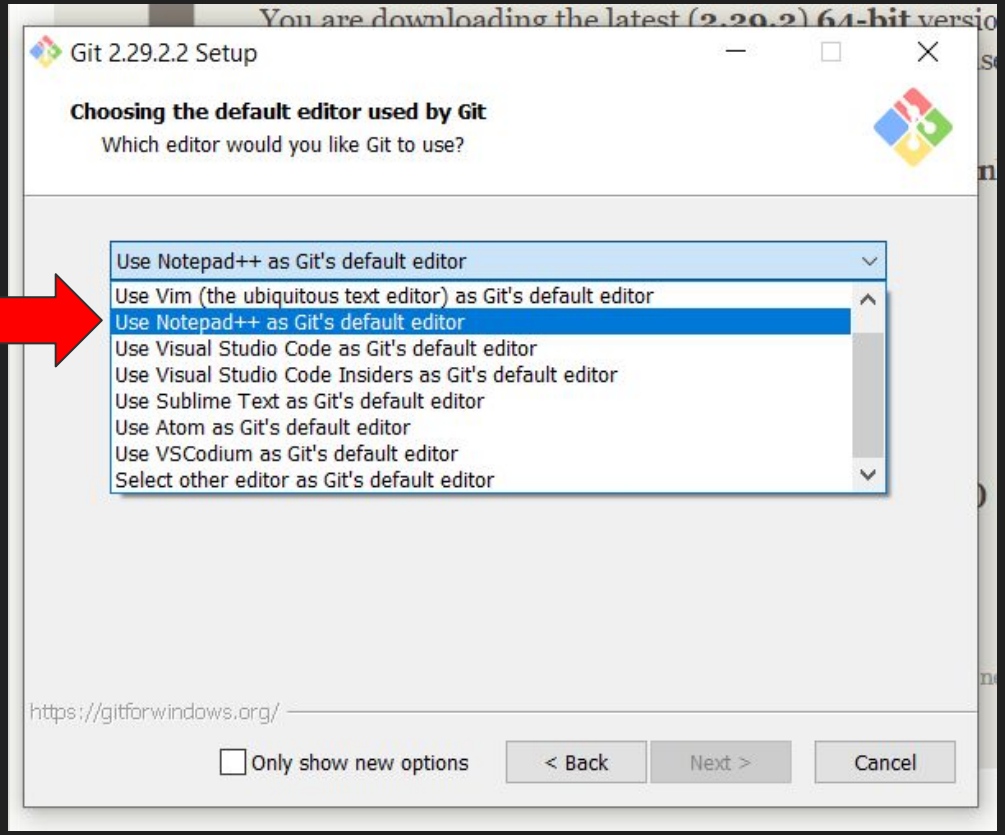
Installing git (Windows)

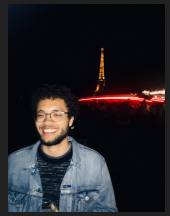




https://git-scm.com/downloads

Installing git (Windows)

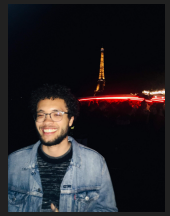


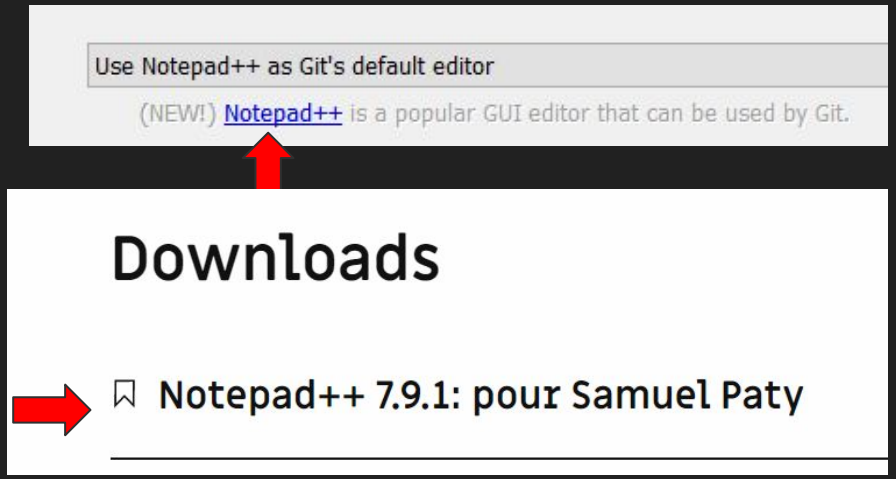


Notepad++ is much easier to use than Vim. In Vim you cannot use your mouse/cursor.

https://git-scm.com/downloads/win

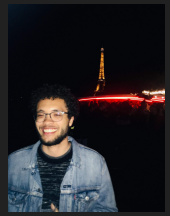
Installing git (Windows)

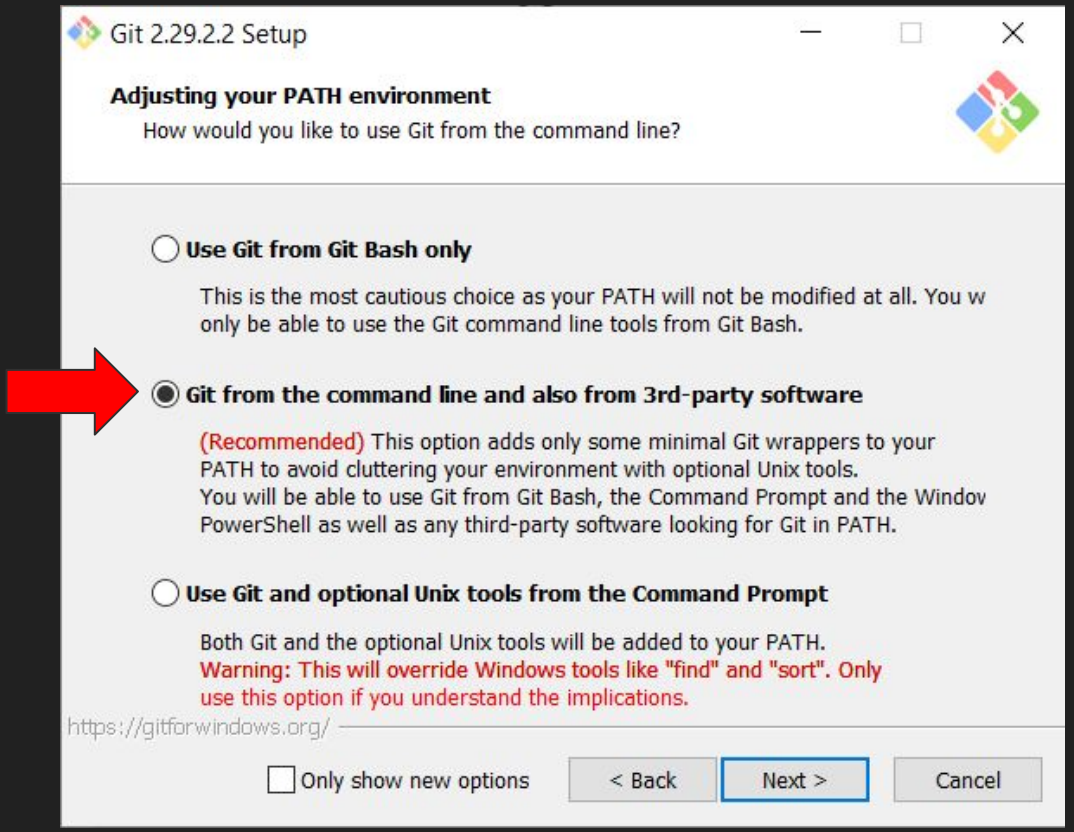




https://git-scm.com/downloads/win

Installing git (Windows)





This option allows you to use git from git bash, Windows cmd, PowerShell and other places!

https://git-scm.com/downloads/win

Installing Git - Mac OSX

If you have Xcode Command Line Tools on your mac, you already have git.

Type **‘git --version’** and press enter

If you see similar output, a git version number, then you already have git, and you can skip the next steps!

Installing Git - Mac OSX - Install a Disk Image

Navigate to: https://sourceforge.net/projects/git-osx-installer/

Click on the link to download the latest disk image file (it will be at the top of the list)

Installing Git - Mac OSX - Check Installation

Check to make sure your install is successful by typing the following command:

Type **‘git --version’** and press enter

If you see similar output, congrats, you have git!

Issues setting your PATH? Get help here

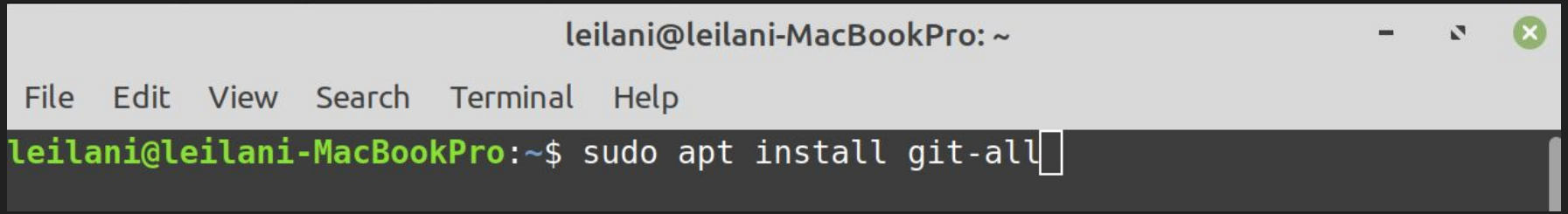
Installing Git - Linux - Debian-based distros

If you have a Debian based Linux distribution, use the following command:

Debian ditros:

Ubuntu, Linux Mint, MX Linux, Deepin Linux, Kali Linux, PureOS.

Type **‘sudo apt install git-all’** and press enter



Installing Git - Linux - RPM-based distros

If you have an RPM-based Linux distribution, use the following command:

RPM distros: Fedora, RHEL, CentOS.

Type **‘sudo dnf install git-all’** and press enter

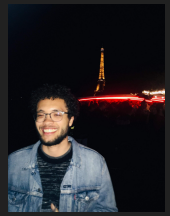
```
:~$ sudo dnf install git-all
```

Installing Git - Linux - Check Installation

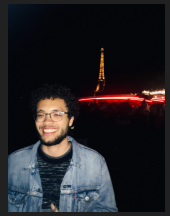
Type **‘git --version’** and press enter

If you see a version number, your install was successful!

First time set up

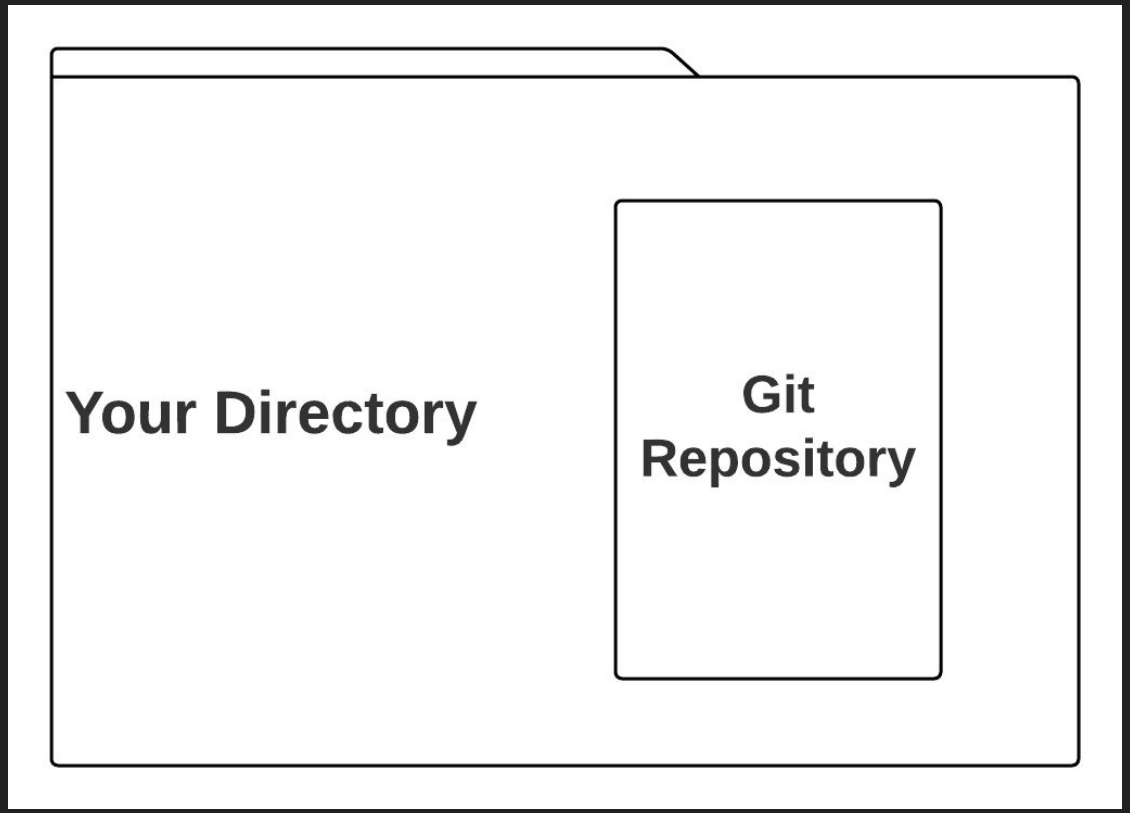


Initializing a Local Repository

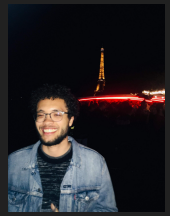


```
michael@Michael123Laptop MINGW64 ~/Desktop/Projects/git_demo
$ git init
Initialized empty Git repository in C:/Users/micha/Desktop/Projects/git_demo/.git/
```

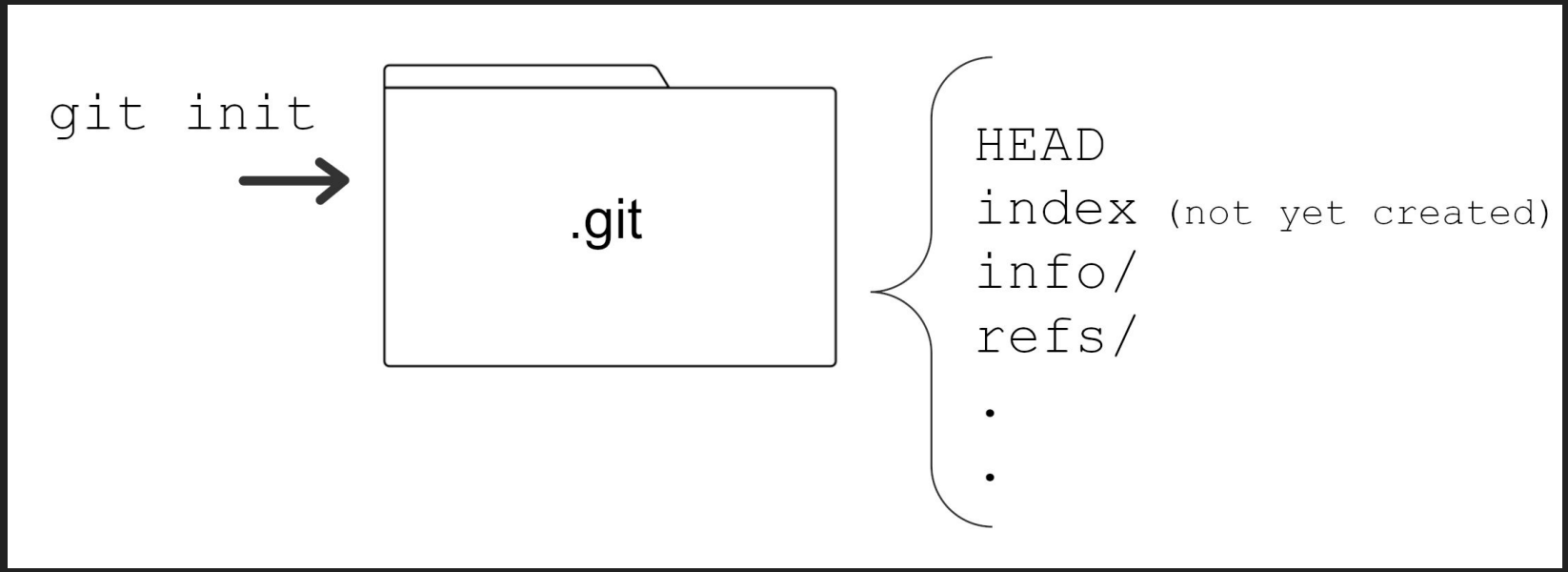
So… what is a repository, really?



What makes the repository?

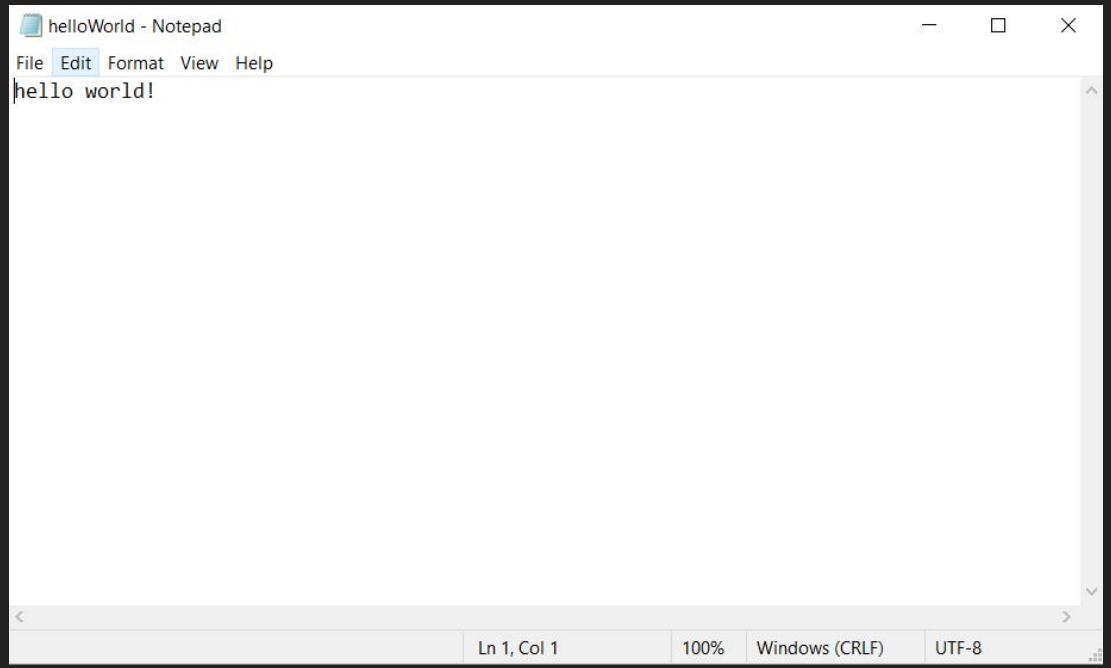


When you run git init, git creates the repository which is a hidden folder in your working directory. This folder contains important files and folders that Git uses to operate.



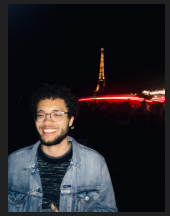
creating & editing files through git bash

```
michael@Michael123Laptop MINGW64 ~/Desktop/Projects/git_demo (master)
$ start helloWorld.txt
```

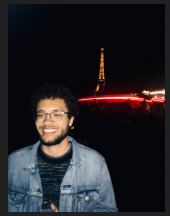


Staging and Tracking Files

Type ‘**git status**’ and press enter

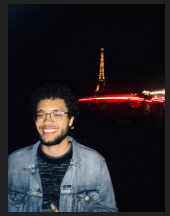


```
michael@Michael123Laptop MINGW64 ~/Desktop/Projects/git_demo (master)
$ git status
On branch master

No commits yet

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

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



```
michael@Michael123Laptop MINGW64 ~/Desktop/Projects/git_demo (master)
$ git status
On branch master

No commits yet

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

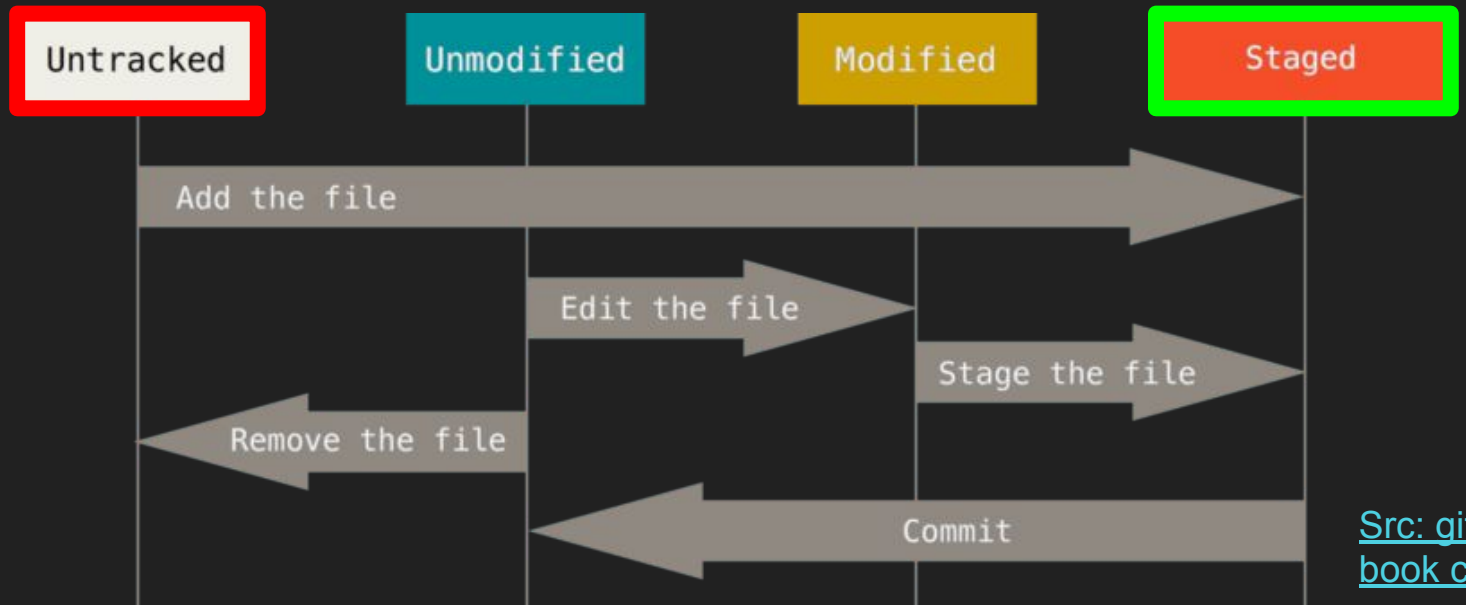
File Staging



The lifecycle of your files in a git repository will follow this flow.

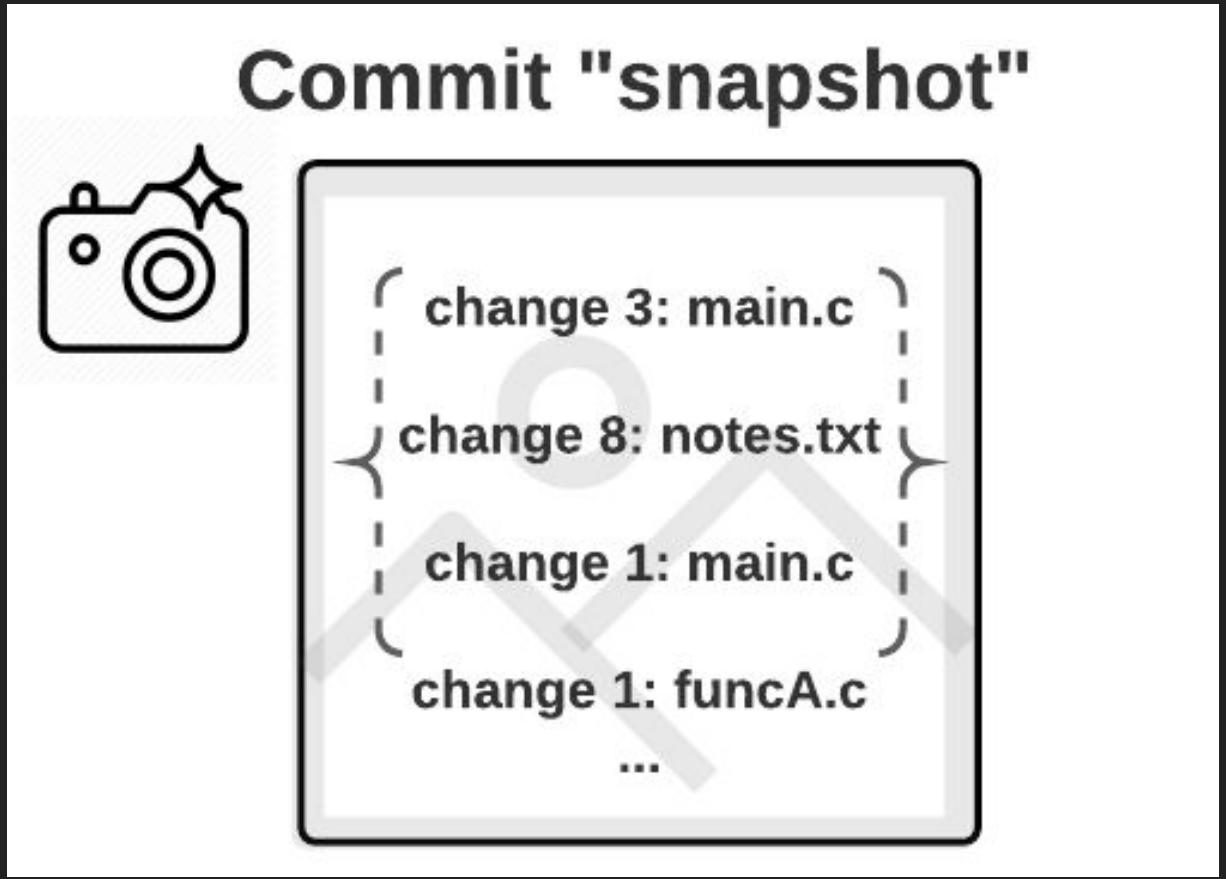
Tracked = included in the repository's previous commits

Staged = added to the current index, ready to be committed in the next commit

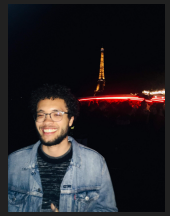


Src: git-scm book chapter 2.2

What is a commit?



Committing

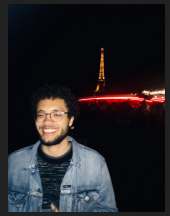


Prepare a **payload** for pushing any changes into remote repository. The commit contains logs file (what have you changed in your code, files).

Command:

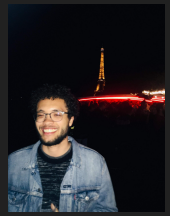
```
$ git commit -m “[your message go here]”
```

Note that every commit require message, -m stand for message.



```
michael@Michael123Laptop MINGW64 ~/Desktop/Projects/git_demo (master)
$ git commit -m"initial commit"
[master (root-commit) 7b95eb3] initial commit
 2 files changed, 1 insertion(+)
 create mode 100644 helloworld.txt
 create mode 100644 history_of_the_world.txt
```

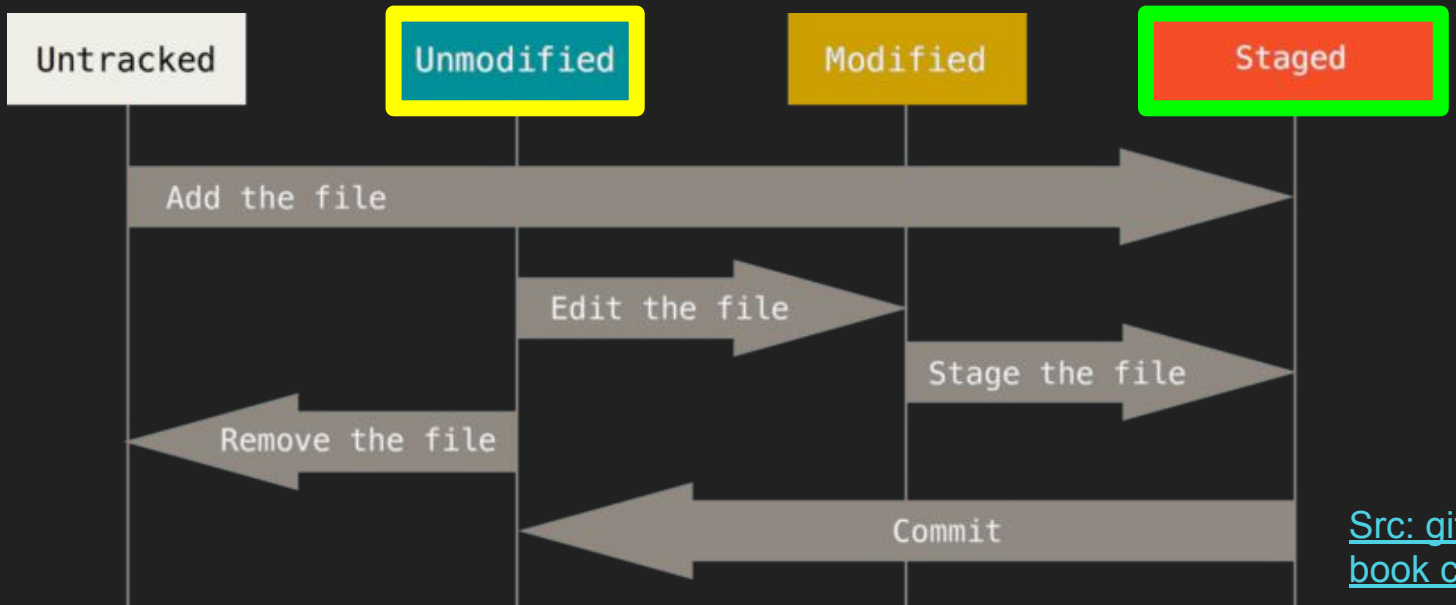
File Tracking and Staging



The lifecycle of your files in a git repository will follow this flow.

Tracked = included in the repository's previous commits

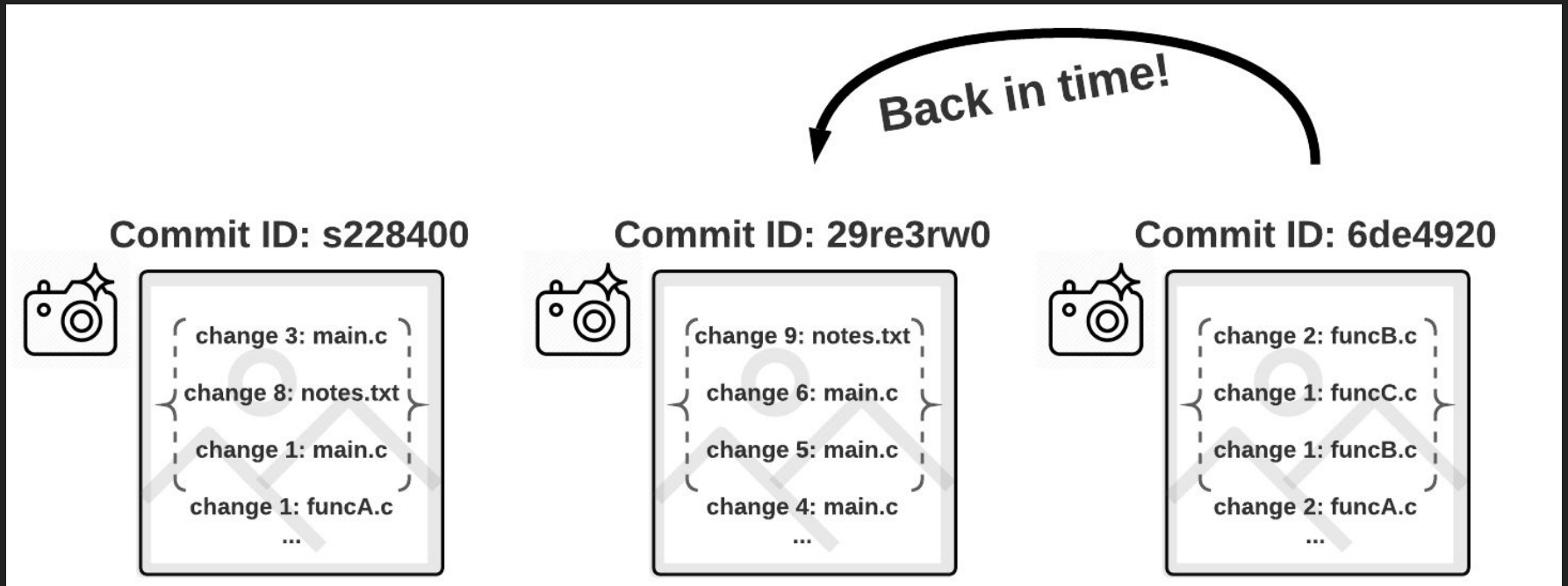
Staged = added to the current index, ready to be committed in the next commit



Src: git-scm book chapter 2.2

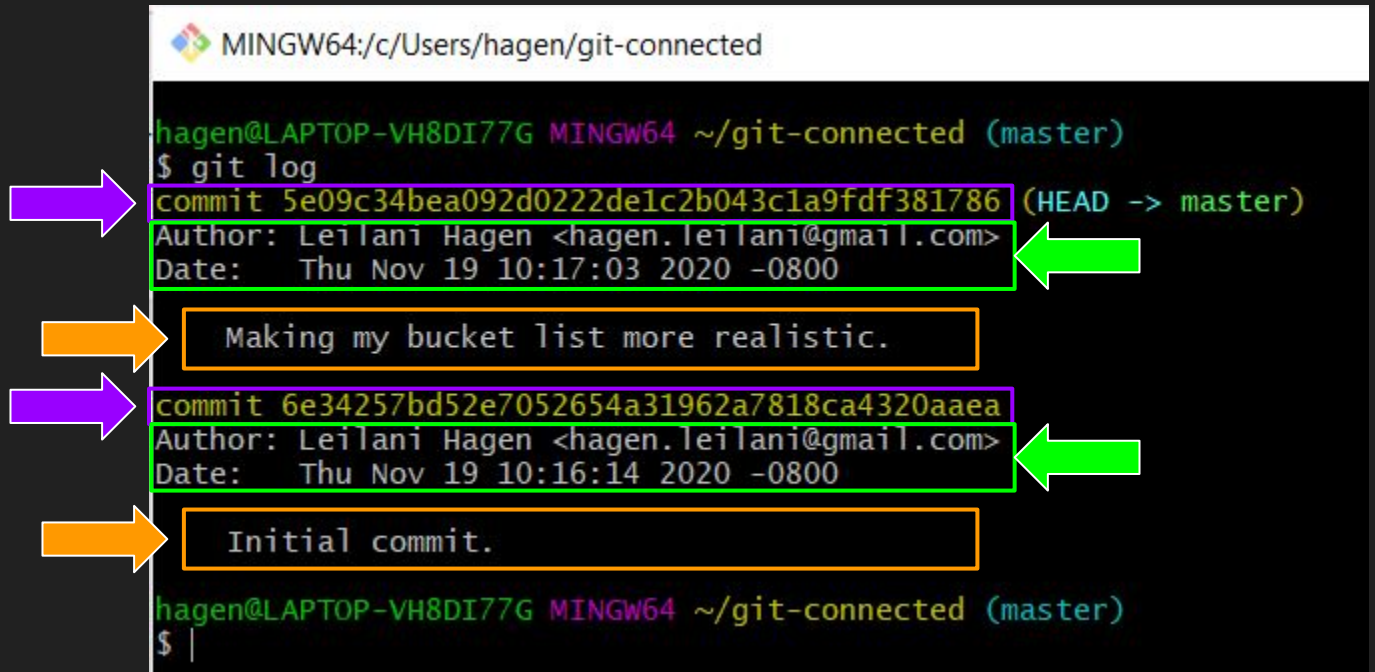
Undoing Things!

- No universal “undo” command
- Go back in time!



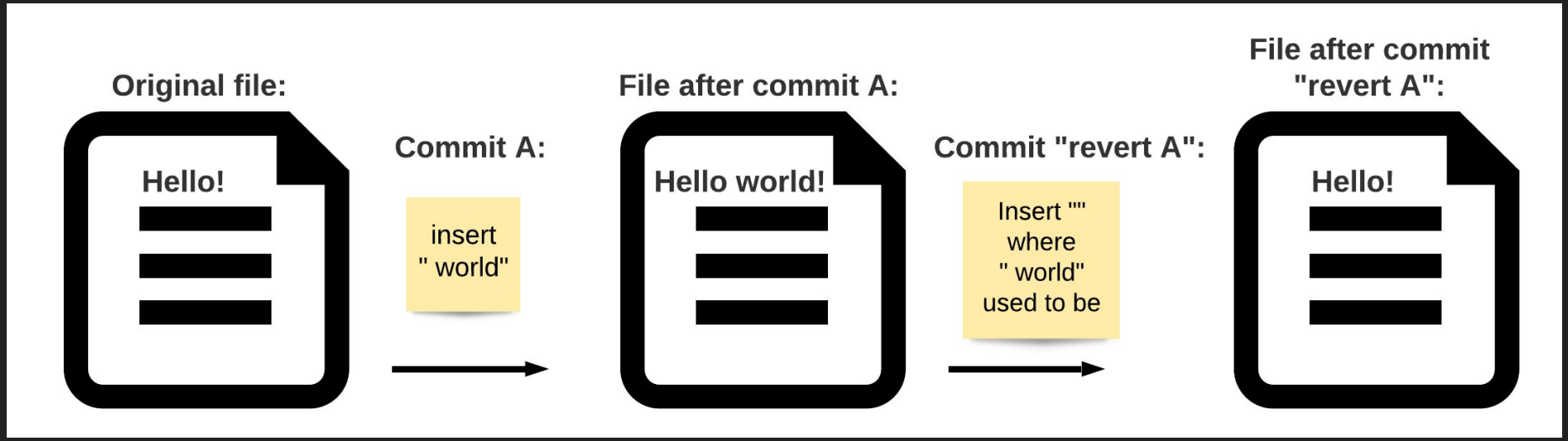
Undoing Things! - Git Log

We can use the git log command to view the git log - a saved list of recent commits made in your repository.



Undoing Things! - Revert

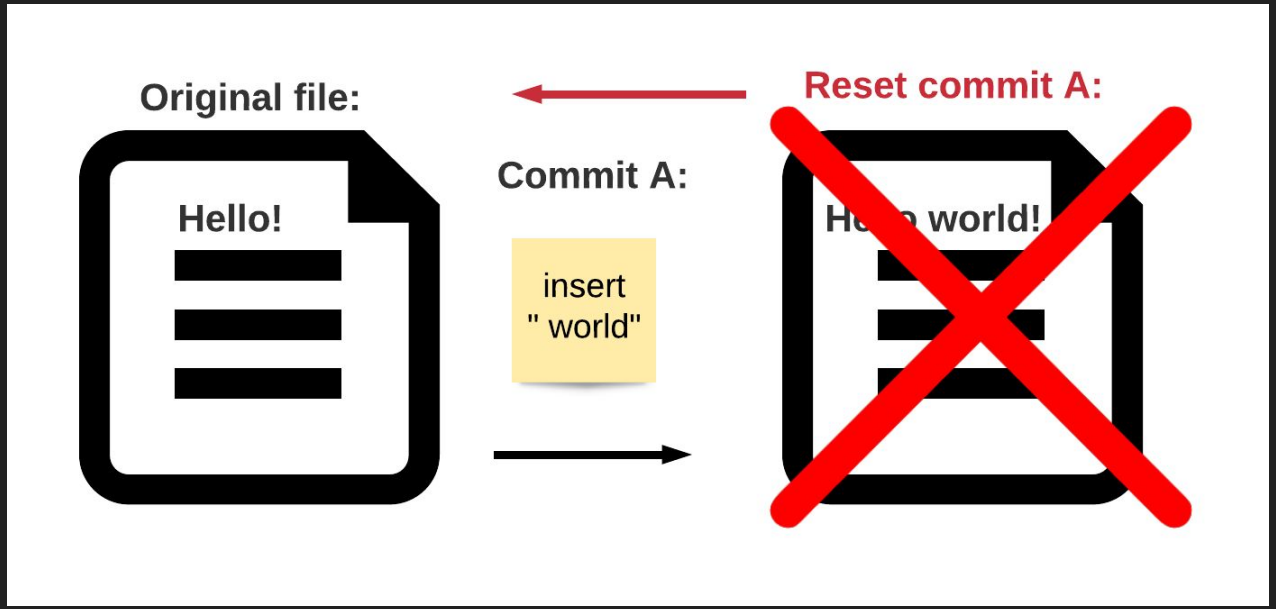
Simplest and safest - commits the inverse of the commit you are reverting.



Read more about “undoing”: https://www.atlassian.com/git/tutorials/undoing-changes

Undoing Things! - Reset

Reset simply wipes gits history of a commit, making it as if the commit never happened and returning to the earlier commit



Read more about “undoing”: https://www.atlassian.com/git/tutorials/undoing-changes

Accidentally Committed too Early?

Just amend! Amend lets you add more staged changes to a previous commit

To amend to your last commit, type **‘git commit --amend’**

and press enter

When Should I Commit? How Often? WHY?!

How often?

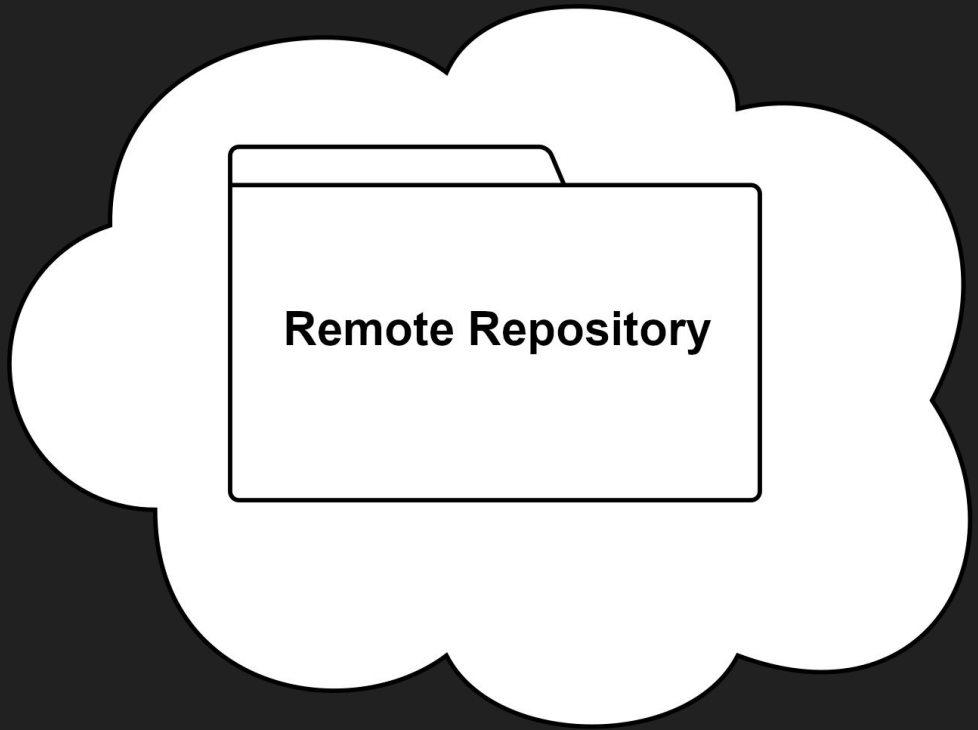
- Every time you reach a point in your project that you want to be able to go back to
- More frequent commits is best

Why?!

- It saves your place in your project
- *Documents* the work in your project! Without you even thinking about it =)

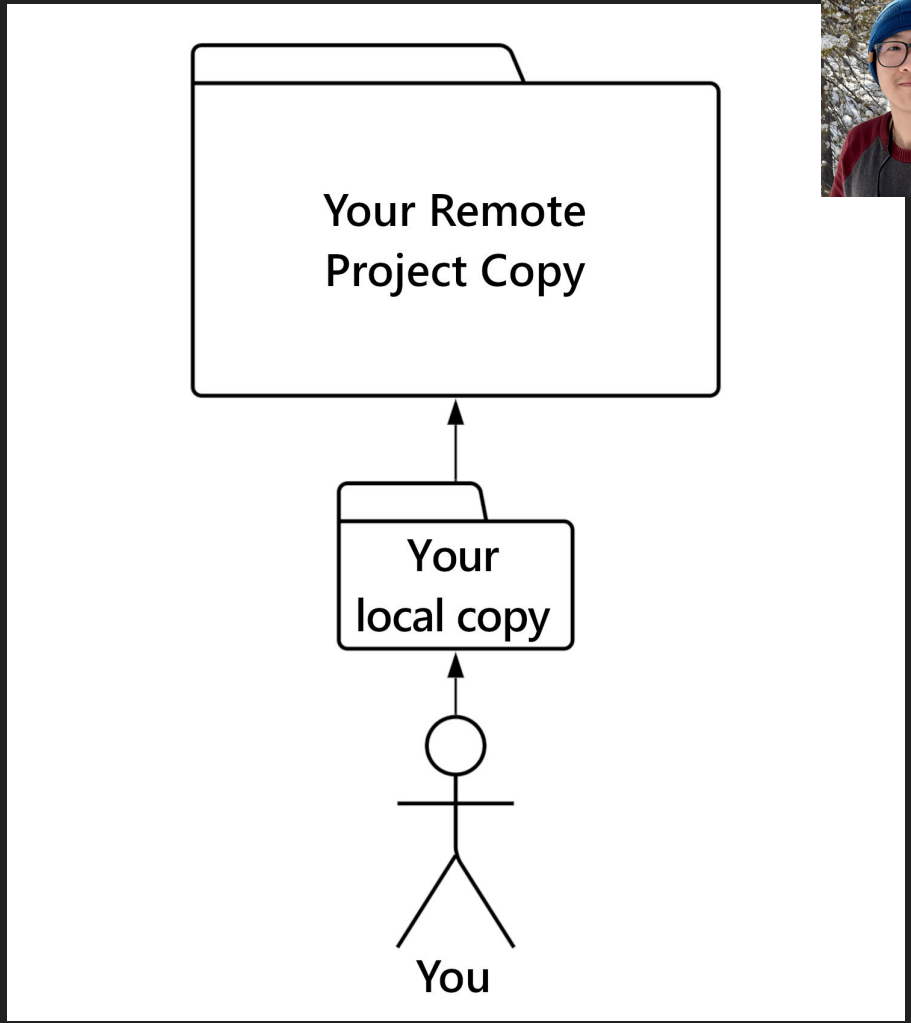
Remote Repositories

Remote repositories are git repositories, just like the one we've just created, that live on the cloud somewhere.



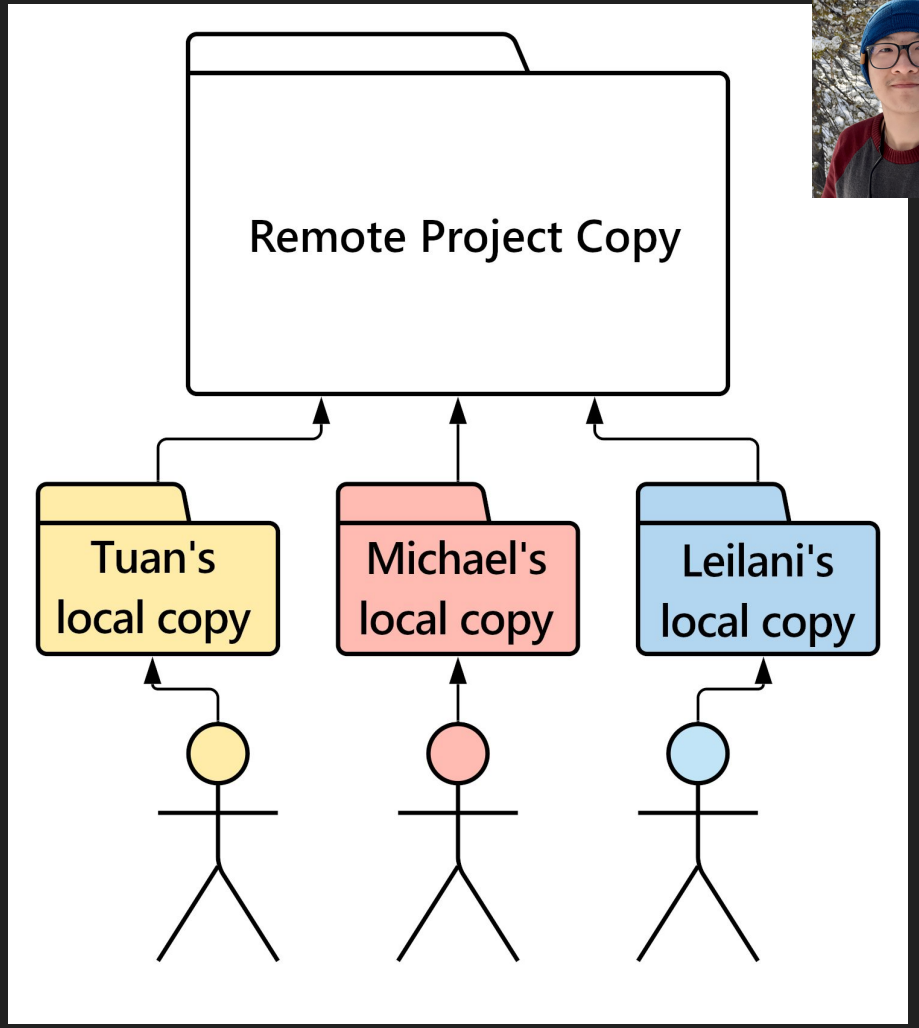
Remote Repositories - Individual

Remote repositories can be used as a way to back-up and version control your own files, without collaboration.



Remote Repositories - Collaboration

Remote repositories are often used as the central point of collaboration between team members or classmates.



SSH Access

In order work with remote repositories, we need to configure a public SSH key.

SSH = Secure Shell network protocol. In short, it lets your computer securely transfer files to your Git remote repository over an unsecured open network. SSH accomplishes this by using a pair of keys, one public key and one private key.

Open git bash

Command: $ ssh-keygen

Src (read more about SSH keys): https://www.atlassian.com/git/tutorials/git-ssh

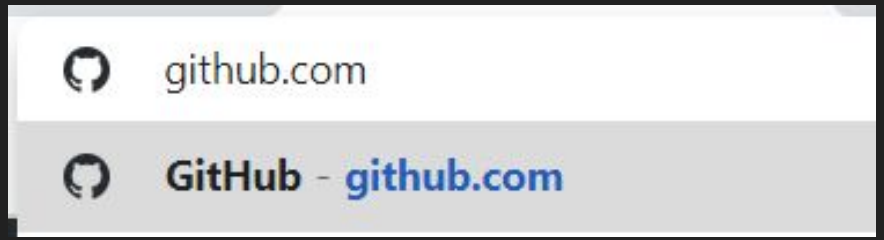
```
nhttu@g7-TuanNguyen MINGW64 ~
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/nhttu/.ssh/id_rsa):
/c/Users/nhttu/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/nhttu/.ssh/id_rsa.
Your public key has been saved in /c/Users/nhttu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:WGeC                                                zp96c0 nhttu@g7-TuanNguyen
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|                 |
|  .++.o+ =o +    |
| =+.+ o.o +      |
|OB   . E S       |
|@==              |
|*o..             |
| o               |
|                 |
+----[SHA256]-----+

nhttu@g7-TuanNguyen MINGW64 ~
```

Creating Your Remote Repository

Once you have configured your SSH keys, you can create a remote and add it to your local repository’s list of remotes. You can then use the remote repo to back up your local repository that you just created.

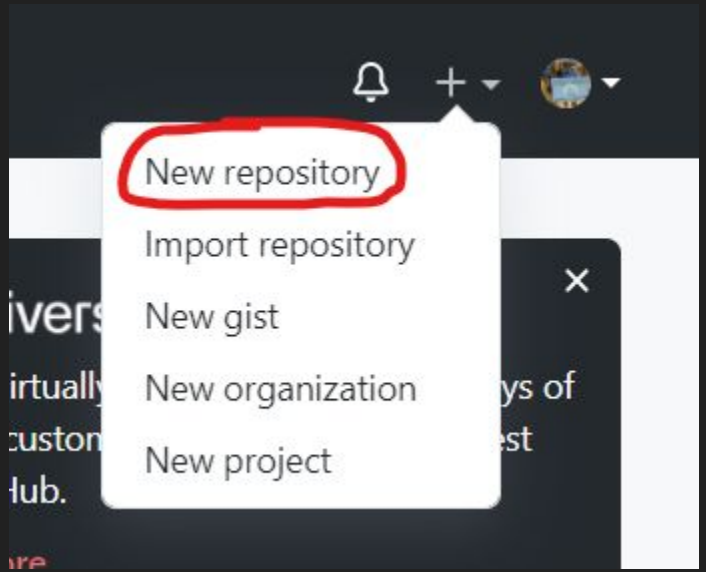
Creating Your Remote Repository

Navigate to GitHub.com and sign in or create an account.



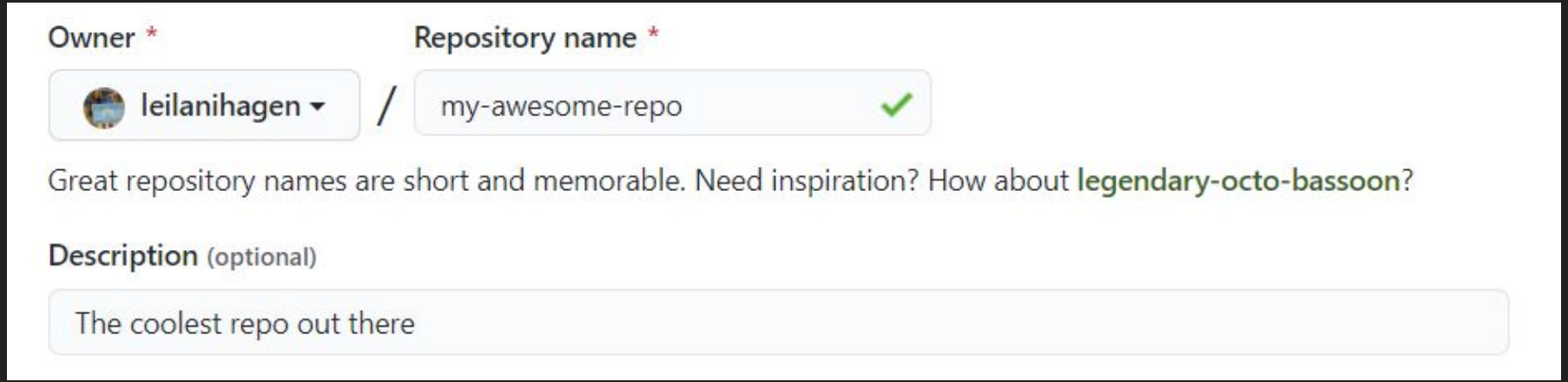
Creating Your Remote Repository

Once you are signed in, find the ‘+’ icon in the top right corner, click and choose ‘New repository’



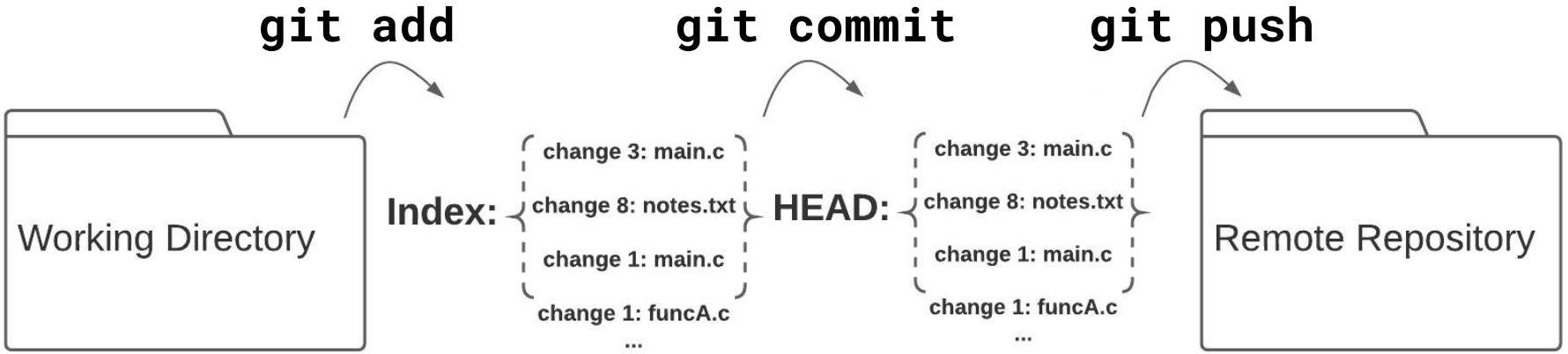
Creating Your Remote Repository - Name

Choose a name (NO SPACES, dashes are common convention), and add a description if you like.



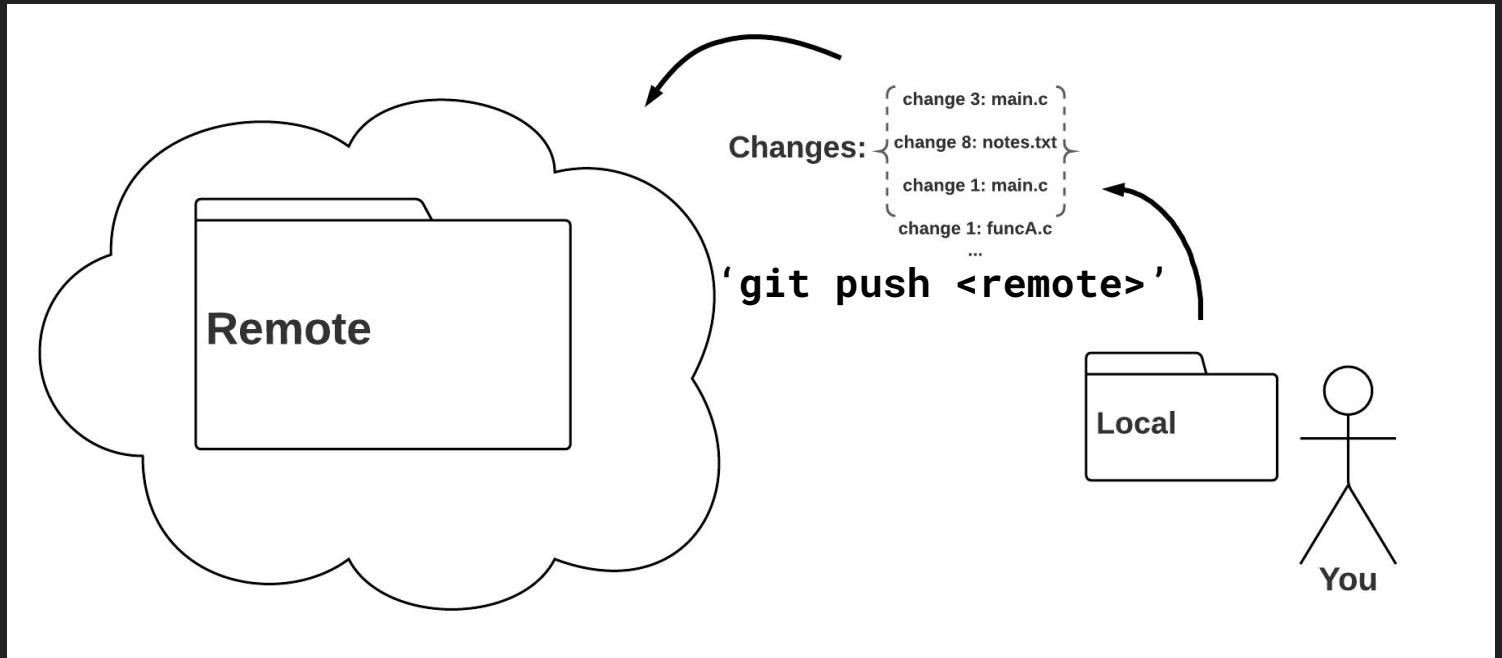
Local-Remote Workflow

The Basic Local-Remote Git Workflow



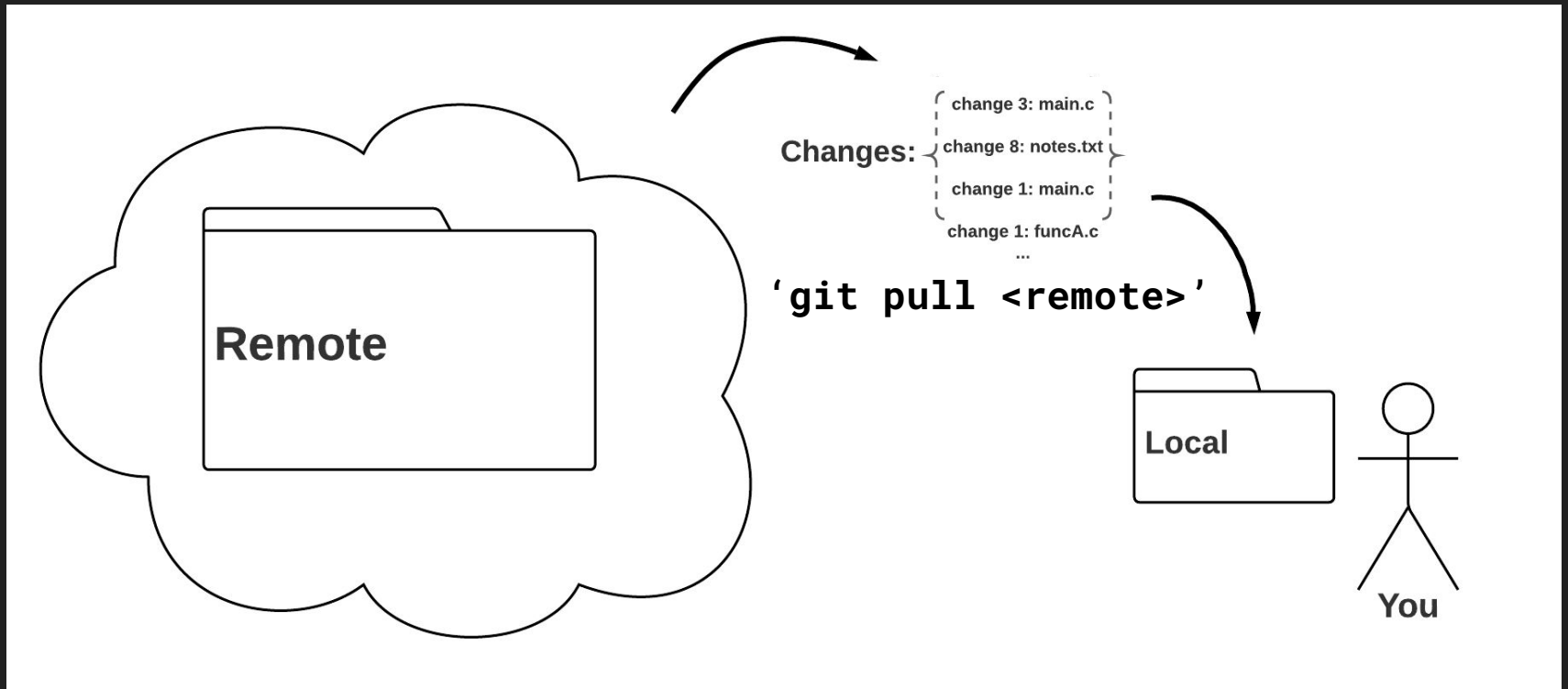
Pushing Changes to a Remote Repository

Pushing is how you transfer a commit (group of tracked changes) from your local repository to your remote repository.



Pulling Changes from a Remote Repository

Pulling is how you update your local repository to be up-to-date with the current state of the remote repository.



Q&A - No Tough Questions!