Git

Andrew Ba Tran

Contents

This is from the seventh chapter of learn.r-journalism.com.

Alright, let's get to git.

After you've installed git, then you can convert a directory on your computer into a repository.

This means its contents and changes can be tracked after this point.

So go back to your terminal.

Creating a new repository

Navigate to the project folder you want to track with git.

Here's a good guide for using the command prompt to navigate to folders on your PC. And a guide for Macs. One shortcut is to drag the folder from Finder onto the terminal icon in the menu and it will go to that folder in terminal.

Once you're in the right folder, type:

```
git init .
```

This tells git to initialize a new repository in the current folder and everything in it, represented by the `.` in the command.

Did you configure git with your name and email yet?

```
git config --global user.email "your@email.com"
git config --global user.name "your name"
```

Committing your work

Now that you've initialized the folder as a repository, you can start logging your work.

This is called "commits".

Though it's not required, a useful first step before commiting your work is to check the status of your git with the command:

```
git status
```

You'll probably get a big list of files under "untracked".

This means that git recogizes that these files exist but isn't tracking them for changes.

You have to use the `add` command.

```
git add .
```

This will add everything in the directory to be tracked.

You can add specific files by typing it out instead of using the `.` period to represent all files.

To be safe, run the add command before every commit to make sure you're not leaving any files behind.

After adding the files, run the status command again.

```
git status
```

Git will tell you that your files have been staged and are ready to commit.

Now, you can log this change with the `commit` command.

You have to include a personalized message, which you add by passing the `-m` command with the string for the message.

```
git commit -m "first commit"
```

That's it!

For future commits, try to communicate what changes were made.

This is good for documentation for if you need to revert back to a previous version.

Note: Get into the good habit of adding and committing frequently.

Great!

It can seem like a lot of work and typing at first but it's the best practice for doing this.

1. add files
2. commit files with a message
3. repeat

You'll get used to it.

Next, we can push these changes online to Github for sharing.