Introduction to version control

INTRODUCTION TO GIT



George Boorman
Curriculum Manager, DataCamp

datacamp

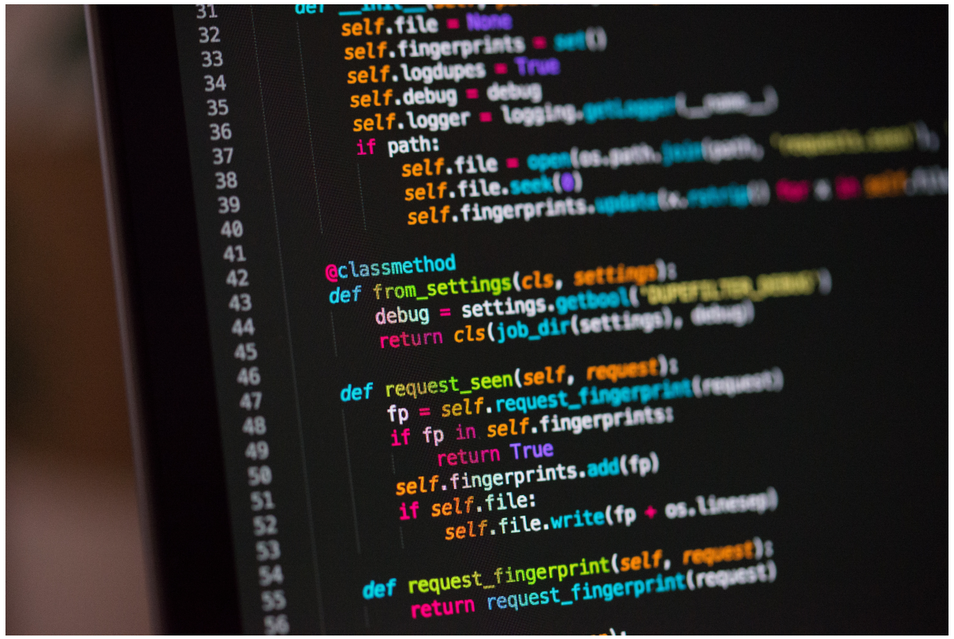
What is version control?

- Processes and systems to manage changes to files, programs, and directories



What should be version controlled?

- Version control is useful for anything that:
  - changes over time, or
  - needs to be shared



What can version control do?

- Track files in different states
- Combine different versions of files
- Identify a particular version
- Revert changes

Why is version control important?



[1] Image credit: https://unsplash.com/@mvdheuvel

Why is version control important?



Introducing Git



- Popular version control system for software development and data projects
- Open source
- Scalable

Benefits of Git

- Git stores everything, so nothing is lost
- We can compare files at different times
- See what changes were made, by who, and when
- Revert to previous versions of files!



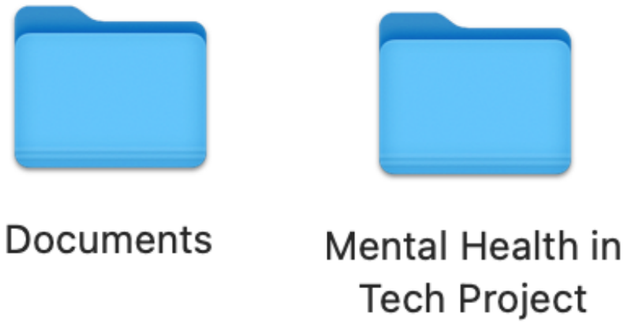
[1] Image credit: https://unsplash.com/@wilsonjim

Using Git

- Git commands are run on the **shell**, also known as the *terminal*
- The shell:
  - is a program for executing commands
  - can be used to easily preview or inspect files and directories



- Directory = folder



Useful terminal commands

```
pwd
```

```
/home/repl/Documents
```

```
ls
```

```
archive    finance.csv    finance_data_clean.csv    finance_data_modified.csv
```

Changing directory

```
cd archive
```

```
pwd
```

```
/home/repl/Documents/archive
```

Checking Git version

```
git --version
```

```
git version 2.46.0
```

Let's practice!

INTRODUCTION TO GIT

Creating repos

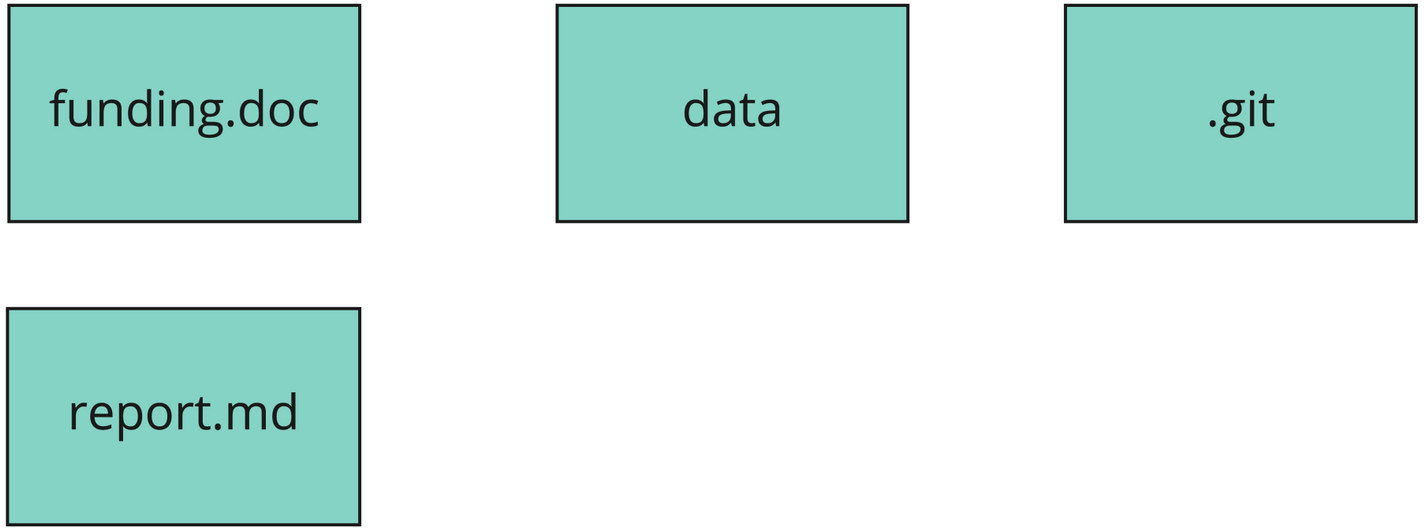
INTRODUCTION TO GIT



George Boorman
Curriculum Manager, DataCamp

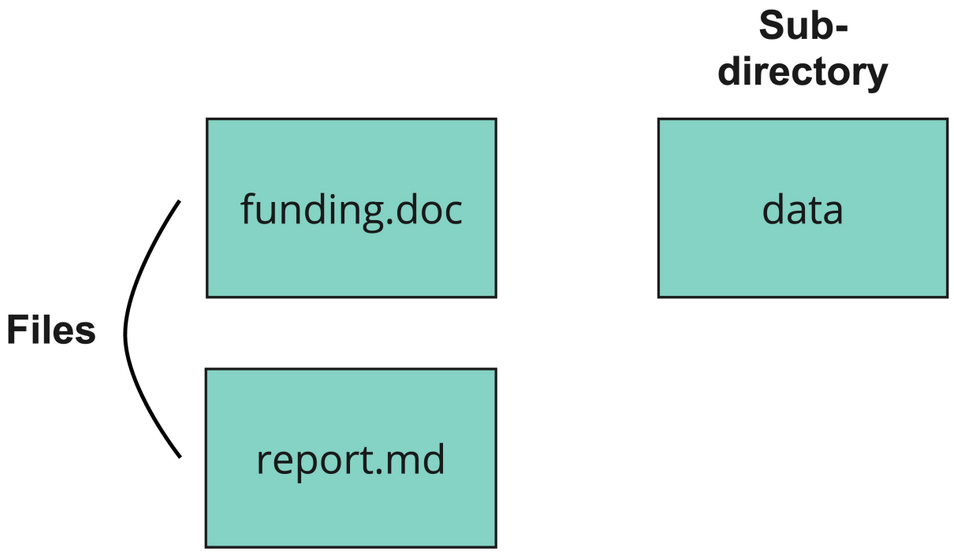
datacamp

Mental health in tech project



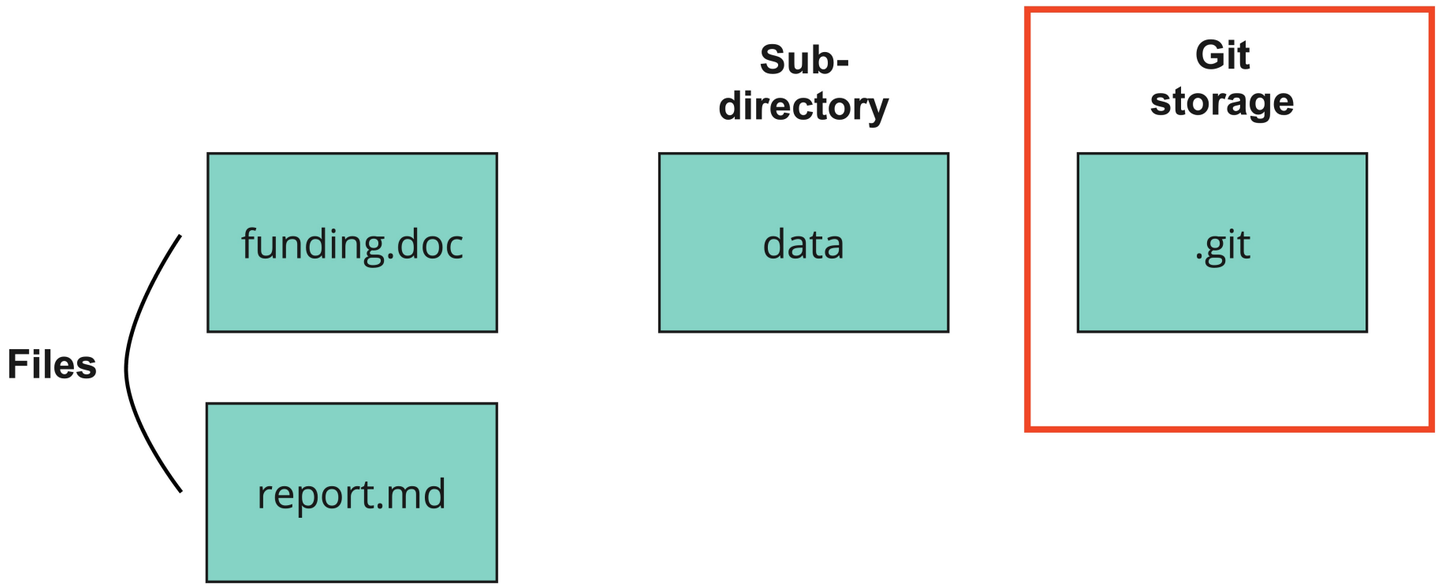
What is a Git repo?

- Git repo = directory containing files and sub-directories



What is a Git repo?

- Git repo = directory containing files and sub-directories, and Git storage



Do not edit `.git` !

Why make a repo?

Benefits of repos

- Systematically track versions
- Revert to previous versions
- Compare versions at different points in time
- Collaborate with colleagues



[1] Image credit: https://unsplash.com/@jasongoodman_youxventures

Creating a new repo

```
git init mental-health-workspace
```

```
cd mental-health-workspace
```

```
git status
```

```
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Converting a project into a repo

- Convert an existing directory into a Git repo

```
git init
```

```
Initialized empty Git repository in /home/repl/mental-health-workspace/.git/
```

What is being tracked?

```
git status
```

```
On branch main

No commits yet

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

        data/
        report.md

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

- Git recognizes there are modified files not being tracked!

Nested repositories

- Don't create a Git repo inside another Git repo
  - Known as nested repos



- There will be two `.git` directories
- Which `.git` directory should be updated?

Let's practice!

INTRODUCTION TO GIT

Staging and committing files

INTRODUCTION TO GIT



George Boorman
Curriculum Manager

datacamp

The Git workflow

- Edit and save files on our computer

- Add the file(s) to the Git staging area
  - Tracks what has been modified

- Commit the files
  - Git takes a snapshot of the files at the point in time
  - Allows us to compare and revert files

Staging versus committing

Staging area



Making a commit



Adding to the staging area

- Adding a single file

```
git add README.md
```

- Adding all modified files

```
git add .
```

- `.` = all files in the current directory and sub-directories

Making a commit

```
git commit -m "Adding a README."
```

```
[main cb33c18] Adding a README.
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
```

- `-m` Allows a log message without opening a text editor
- Log message is useful for reference
- Best practice = short and concise

Let's practice!

INTRODUCTION TO GIT