Center for
Data Science

Lab 1: Git, GitHub Classroom and SQLite

What is Git and Version Control?

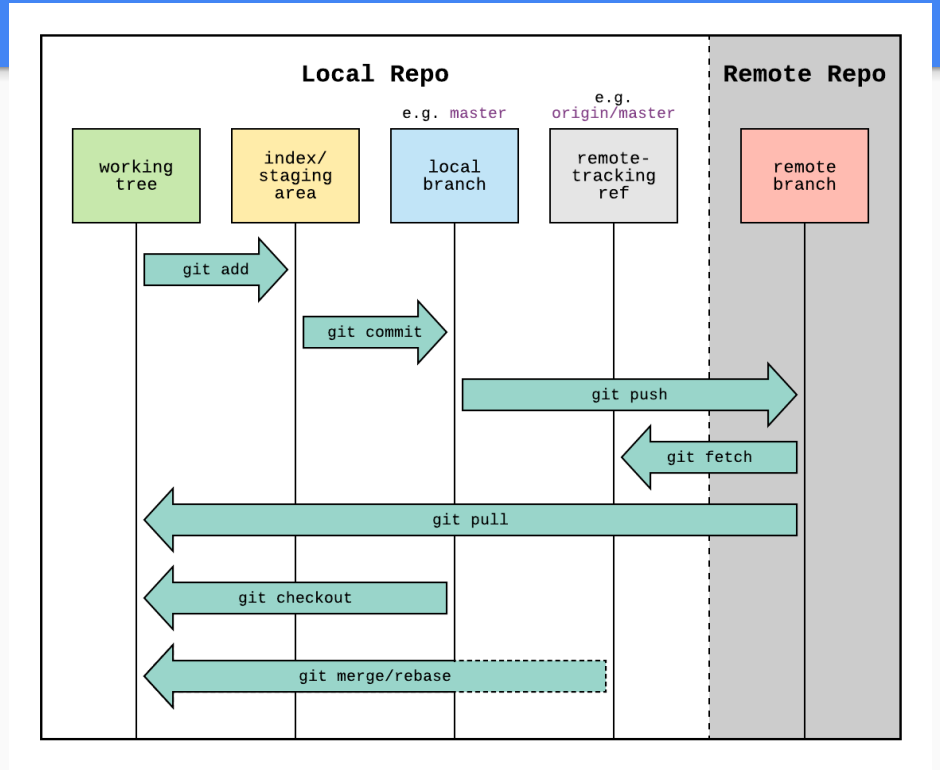
- **Version Control Systems (VCS)**
  - tools that store different versions of project files
  - revert to earlier versions
- **Git** is a **distributed** VCS
- **GitHub** is a website that hosts git repositories
  - We'll use it to distribute and collect lab assignments!

Setting up git

- Go to https://github.com/
  - Sign Up/Create new account if you don't already already have one.
- Mac and Linux users generally have git already installed on their system
- For Windows users follow instructions on this link:
  - Windows Subsystem for Linux: https://docs.microsoft.com/en-us/windows/wsl/install-win10
- OR install git on Windows:
  - https://help.github.com/articles/set-up-git/#setting-up-git

Cloning a repo(sitory)

- In order to work with a repository, you first need to make a copy that runs on your computer.
- Making this copy is called “cloning”.



Basics (Making changes to a repo)

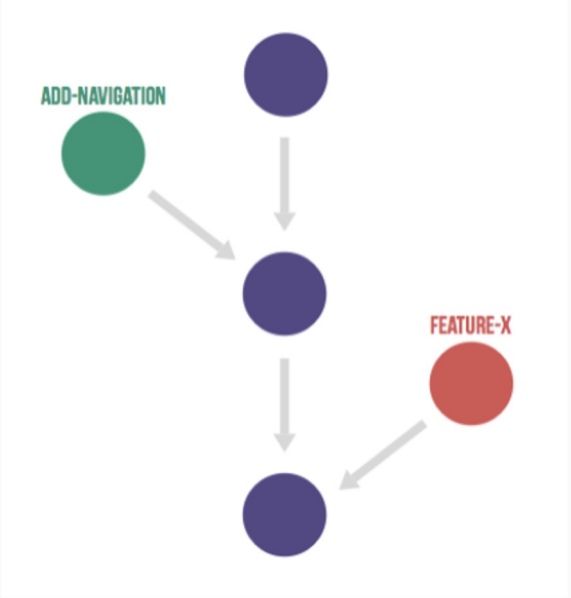
- **Pull**: Use **git pull origin main** to pull any latest changes from the forked repo to your local copy.
- **Status**: Use **git status** command to see the staged (shown in green) and un-staged (shown in red) files in your local repository.
- **Staging**: Use **git add <filename>** to stage a changed file for commit
- **Commit**: Use **git commit -m “<your message here>”** to commit the staged files.
  - Keep your message short, descriptive and specific.
- **Push**: Use **git push origin main** to push all the changes made locally to the origin.

DEMO: Navigating Github classroom

DEMO: Pull, push & commit

Basics (Branching & Merging)

- **Branching**:
  - **git switch –c <branch name>** to create a new branch
  - **git switch <branch name>** to switch to a different branch
- **Push**:
  - **git push origin <branch name>** to push any changes made on this branch.
- **Merging**:
  - **git merge <branch name>** to merge changes in <branch name> to your current branch.



Basics (logging)

- **Log**: Use **git log <options>** to view the history of changes

- Different options, e.g.:
  - **git log --help**
  - **git log --decorate --all**

More references available: https://swcarpentry.github.io/git-novice/

Potential for confusion (empty folders)

- Git is *not* just like the folder structure on your computer, but it looks like it, at first glance, which is problematic.
- In Git, you cannot just create a new, empty folder.
- The file is the basic unit of organization in Git, as its primary purpose is content tracking.
- So everything starts with a readMe file (even if it is empty).

Homework 1 Assignment

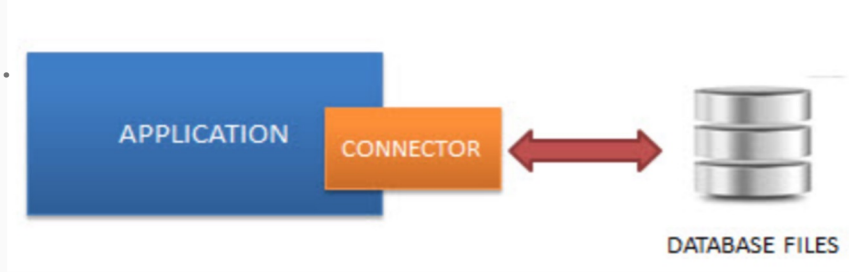
- On Brightspace
  - Assignments → Homework 1
- Instructions for lab assignments are provided in the README on Github classroom.
- Note: This is the *only* homework that you have to do by yourself (not as part of a small group). The reason for this is that logistically, we need to make sure that you *have* a github account and can access the Github classroom account of the class with that account.
- **Submit the assignment by pushing to github**. That's it!
  - Check the file contents / commit history through github.com to verify your changes
- Submission due date for this assignment is **2024-02-05**.

NYU | Center for Data Science

Now for something completely different: SQLite

What is SQLite?

- SQLite is a “lightweight” version of SQL
- SQLite is serverless: DB is stored in a single file
- Transactions in SQLite are fully ACID-compliant
  - Atomic, Consistent, Isolated, and Durable.



Reference: http://www.sqlitetutorial.net

What is the difference and how is it “lighter”?

- SQLite does not need to be ”configured” as a server like MySQL.
- Serverless- no separate server process is needed
- Transactional SQL database engine
- Stable Cross-Platform Database File
    - can be written/ copied on different machines with different architecture

Basic Commands

Attach Databases
Create Table, Insert Into the table
Retrieve Information (SELECT)
Filter data (WHERE, LIMIT, BETWEEN, IN)
Group By Data with HAVING clause
Sort Data ASC/DESC
Joins (LEFT, OUTER, INNER, CROSS)
Alter Table, Update Table
Drop Table, Delete Table
Aggregations
Transactions

Order of Execution

1. From (Join + On)
2. Where
3. Group By
4. Having
5. Select
6. Order By
7. Limit

Python and SQLite

To perform any operation on a SQLite database via Python's sqlite3 module

A connection needs to be opened to a SQLite database file

Then you can **execute()** queries through the connection object

```
import sqlite3
conn = sqlite3.connect('music-small.db')

t = (1990,)  # note that this is a tuple!

for row in conn.execute("""SELECT count(*) FROM artist
                    WHERE artist_active_year_begin=?""", t):
        print(row)
```

DEMO: Connecting and query execution

Do’s and Don’ts

Don’t use this as this is insecure.

```
value = ‘1998’
conn.execute(“SELECT * FROM tracks WHERE year = ‘%s’ “ % value)
```

Always do this:

```
value = (1998,)
conn.execute(“SELECT * FROM tracks WHERE year =?”, value)
```

Any questions?

- Anything at all!
- Don’t be shy!