Managing Projects with Git

(and other command-line skills)

Dr. Chris Mayfield

Department of Computer Science
James Madison University

Feb 17, 2022

GitHub

Part 1: Group Repository

What is Git/GitHub?

Git is a version control system

GitHub is a social network

Tutorials:

- https://guides.github.com/activities/hello-world/
- https://kbroman.org/github_tutorial/

Cheat Sheet:

- https://education.github.com/git-cheat-sheet-education.pdf

Setup (IMPORTANT)

`git config --global user.name "firstname lastname"`

- select a name that is identifiable for credit when reviewing version history

`git config --global user.email "email address"`

- select an email address that will be associated with each history marker

`git config --global color.ui auto`

- automatic command line coloring for Git for easy reviewing

GitHub simplified

1. Clone your repository (one-time setup)
   - `git clone https://github.com/cs374/`*`teamname`*`.git`
2. Make changes to the files
   - Using your favorite editors, dev tools, etc.
3. Add new files, commit changes
   - git add newfile.txt
   - git commit -m "this is what I changed"
4. Pull changes from team members
   - `git pull`
5. Push your changes to the repository
   - `git push`

Git quick reference

Typical workflow

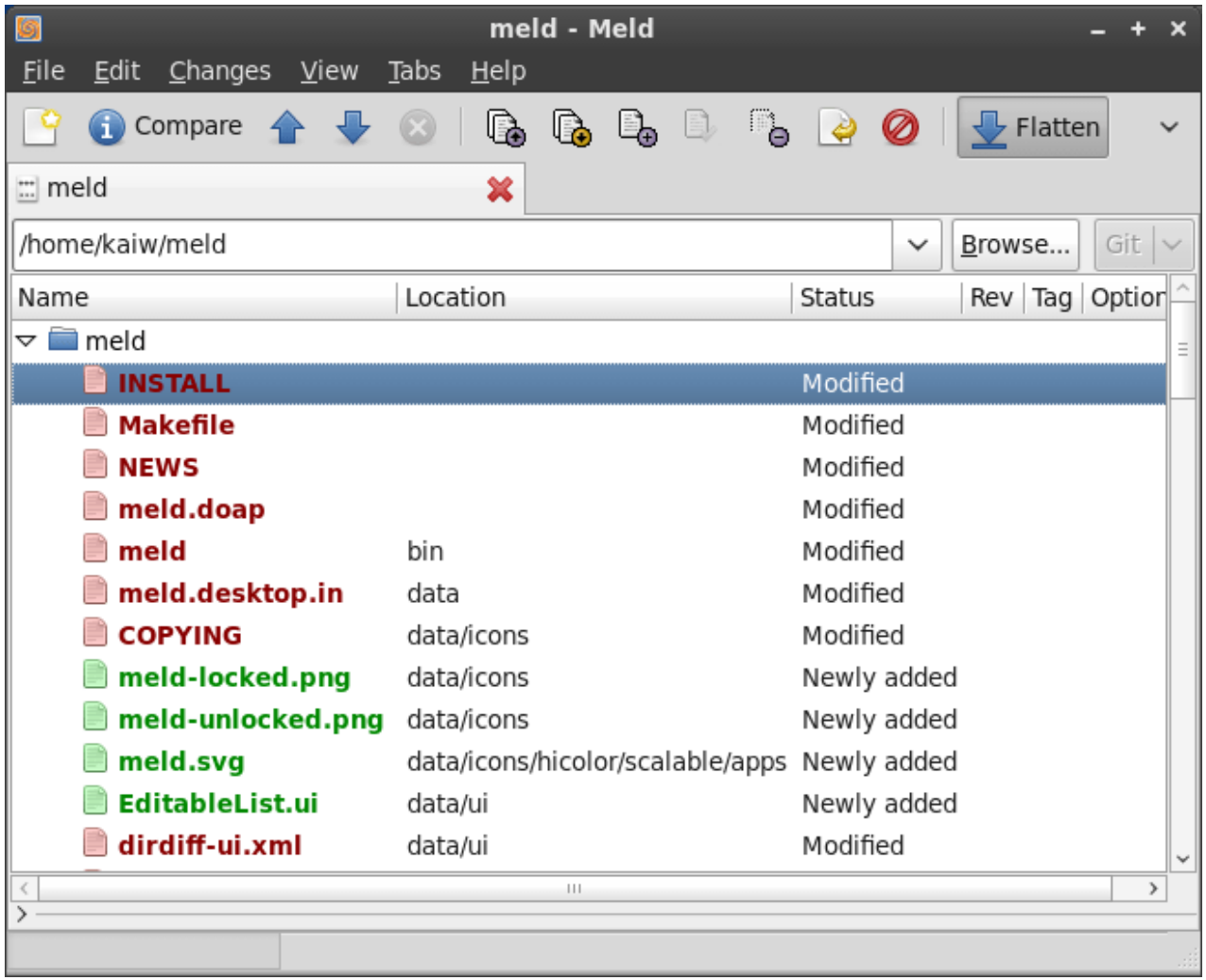
`git pull`	*merge in the latest commits*
`...`	*make changes to your files*
`git status`	*what files have I changed?*
`git diff`	*show changes I have made*
`git commit -m "message"`	*save changes to repository*
`git push`	*upload my latest commits*

Other commands

`git add FILE`	*add new FILE to repository*
`git rm FILE`	*remove FILE from repository*
`git mv SRC DST`	*move/rename in repository*
`git checkout FILE`	*undo local changes to FILE*
`git help`	*show a list of git commands*

Using meld with git

In the Linux lab, use `meld .` instead of `git diff`



Part 2: Group Database

Your group database

(My example group name is `absent`)

To connect using psql

- `psql -h data.cs.jmu.edu absent`

Remember to change the owner

- `CREATE TABLE person (pid integer, name text);`
  - `person` is owned by current user by default
- `ALTER TABLE person OWNER TO absent;`
  - `person` is now owned by the entire group

GP2 will use psql to import data

```
psql -c "\copy person FROM person.csv WITH CSV HEADER" absent
```

Input and output

Pipelining: *program1* | *program2*

- make the output of p_1 the input of p_2
- can be used to chain multiple processes
  - `grep 2012 movies.csv | sort | head -n 20`

Redirection: *program* < *input_file* > *output_file*

- read a file instead of the keyboard
- write a file instead of the terminal
  - `psql absent < create.sql > results.txt`

These operators work on Linux, Mac, and Windows!

Environment variables

Instead of typing:

- `psql -h data.cs.jmu.edu -U mayfiecs absent`
  (and then typing your database password)

Define variables:

- `export PGHOST=data.cs.jmu.edu`
- `export PGUSER=mayfiecs`
- `export PGPASSWORD=123456789`
- `export PGDATABASE=absent`
- `psql` (with no arguments)

(On Windows, use `set` instead of `export`)

https://www.postgresql.org/docs/11/libpq-envars.html