# GitHub Exercise

JD Kilgallin

CPSC:480

09/12/22

# Notes

- Quiz Wednesday a
- Quizzes are not weighted; grades are averaged regardless of points used in each. E.g. 1/1, 4/8, & 0/10 give 75% after dropping lowest.
- See blackboard announcement on some sample projects in my GitHub account that you could start from for a portfolio project.
- It is acceptable to have work-in-progress projects in your portfolio.
- Email me if you're not sure if something belongs on GitHub.
- Still missing 5 GitHub usernames. If you got an email about it, get that to me tonight.

# Learning Objectives

- Navigating GitHub
- Creating a GitHub repository
- Working with Git repositories on the command line
- Making, sharing, and merging changes in Git
- Branches and forks of Git repositories

# High-level plan

- You will need: Partner, computer, GitHub account, Git client software.
- Create a repo, commit some data
- Fork partner's data, edit+commit, issue pull request back to partner.
- Make some conflicting changes and resolve the conflicts.
- Create and merge branches.
- Use git commands to understand repository state and revert changes.
- View commit history and make changes from an old snapshot.
- Use markdown to create a writeup
- Create a release on GitHub

# Part I – Initialization

- Open laptop with Git installed and create parent directory for projects (e.g. C:\Users\me\github).
- Create a public GitHub repository named "SWEF22-Ex-1-<UAid>".
- Open git command line and go to project parent directory.
- Clone a local copy ("git clone https://github.com/<user>/SWEF22-Ex-1-<UAid>") and make sure you're in the repository directory.
- Create file "data.txt" in local repo folder and put your name in.
- Add to files tracked by Git ("git add data.txt" or "git add *").
- Create an initial commit ("git commit -a -m "Initial commit").
- Push change to GitHub ("git push").
- Set author and GitHub credentials as needed.

# Part II – Collaboration

- Open a browser to https://github.com/<partner-username> /SWEF22-Ex-1-<partner-UAid> while logged in to your account.
- Click “Fork” in top right, accept defaults, and click “Create fork”.
- On your machine, go back to github directory, outside of repo,
- Create local copy of fork (“git clone https://github.com/<**your**-username>/SWEF22-Ex-1-<**partner**-UAid>”) and go in new repo.
- Edit data.txt and add today’s date **above** partner’s name.
- Commit and push to your fork on GitHub.
- Open your fork in a browser and click on “Open pull request” under “Contribute”, then “Create pull request”.
- Wait for your partner to complete step 7 before starting step 9.
- Go to your own repo, click “Pull requests” tab, find and open your partner’s PR, and click “Merge pull request”, then “Confirm”.

# Part III – Conflicts

- On your local machine, “git pull” **both** repos to current state.
- Edit data.txt in both repos’ copies and add your UAid at bottom.
- Commit & push the change **only to your fork of *partner’s* repo**.
- *Issue another PR to your partner & accept the one they issue you.*
- *Wait for your partner to complete step 4 before starting step 6.*
- Go to **your** repo, commit your local change and try to push.
- Note the failure, then git pull to download the commits on GitHub that come from your partner’s pull request.
- Open data.txt and view the conflict to be resolved. Edit the file to contain your UAid, then your partner’s.
- Commit the merge and push to GitHub.

# Part IV – Branches

- Create branch in your repo named "part-4” (“git branch part-4).
- Set the new branch as the active branch (“git checkout part-4").
- Rename data.txt to authors.txt (“git mv data.txt authors.txt”).
- Commit the change to this branch (**do not push**).
- Switch to main branch (“git checkout main”) and note that the file is again called data.txt.
- Add your favorite Pokémon (eg "Pikachu") at the end of data.txt.
- Commit the change (**do not push**).
- Merge part-4 branch (“git merge part-4").
- Commit and push the change. Note that the file is called authors.txt *and* has the Pokémon you added while the file was called data.txt.

# Part V – Navigation

- View the list of branches and the active branch (“git branch”).
- Delete data.txt (using “git rm data.txt” will stage the deletion for the next commit, any other method will not; doesn’t matter here)
- View the changes to the current state (“git status”).
- Restore data.txt (“git restore data.txt”).

# Part VI – History

- Find the commit where you added the date in Part II (“git log”) and note the first 4 characters of the hash for this commit.
- Set your local repo state to this commit (“git checkout <hash>”).
- Create a new branch “part-6” (“git branch part-6").
- Add your favorite movie at to data.txt. Commit but don’t push.
- Check out the main branch.
- Merge part-6. Note the message about conflicts. Edit the file to include the content from Part IV, followed by the movie. Commit and push.
- *Wait for your partner to complete step 6 before starting step 8.*
- *Go to your fork of your partner’s repo and download their changes by clicking “Sync Fork” -> “Update Branch”*
- *Click “Insights” -> “Network” in each repo to see visualization of timeline.*

# Part VII – Documentation and release

- Create README.md file and add to Git.
- Write a summary of what you did in this exercise in Markdown.
- Use at least 5 different formatting or special features (see docs).
  - 1% Extra Credit for every 5 additional features used. List the ones you claim in the README (using a md list counts for 1)
- Commit and push the change.
- *View README on GitHub and verify formatting is correct.*
- *Check partner's repo to ensure you've both followed directions.*
- *Add instructor mention (@Kilgallin) in README.md, commit, push*
- *On the right side of the repo, click "create release", click "Create a tag", enter a tag name "submission", and click "Publish release".*

# Submission and grading

- @mention + release will be considered exercise submission.
- You may make corrections until 4 PM Wednesday.
- Grading will only be based on following directions, approximately 3% per gradable step.
- Do not delete repo until grading is complete and any re-grade requests are resolved.

# References

- Setting up a repository | Atlassian Git Tutorial
- Saving changes | Atlassian Git Tutorial
- Set up Git - GitHub Docs
- Hello World - GitHub Docs
- Resolving merge conflicts
- Basic writing and formatting syntax - GitHub Docs

- *Reading for next lecture: Pressman Ch 7-8*
- *Team project will be assigned next lecture. Start finding a team of 5-6.*