Git Collaboration

INFO 201

Today’s Objectives

Discuss final projects

Practice working in teams with Git/GitHub

Use **GitHub Issues** to track project tasks

Learn how to **rebase** changes

Practice resolving **conflicts**

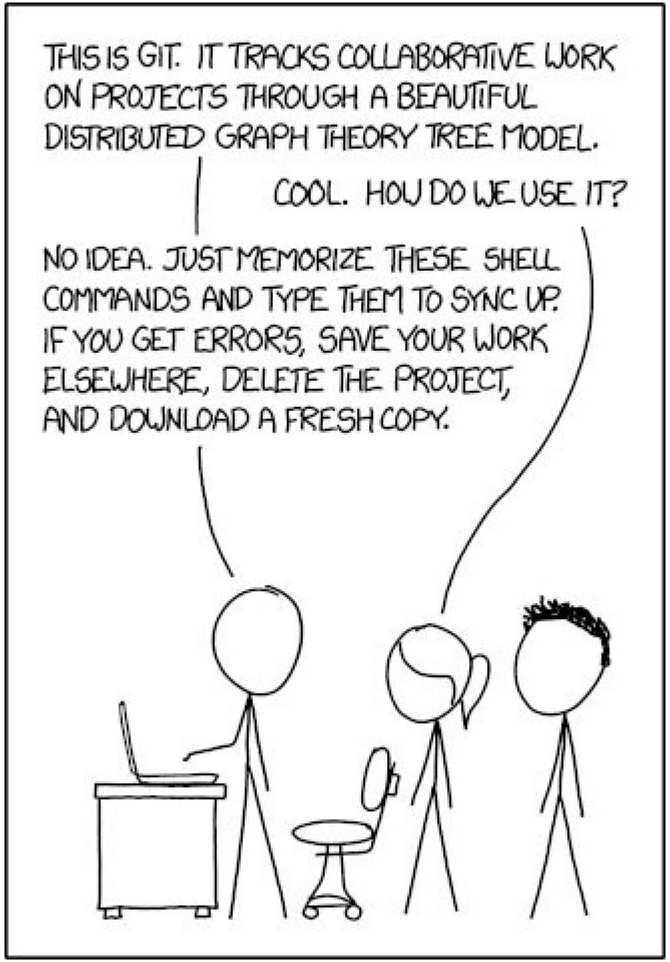
Final Projects

Demonstrate your ability to ***collaboratively write code to work with data*** and share insights.

Some former projects

[US Obesity rates]
[Sleep habits]
[Washington state spending]
[Seattle 911 calls]
[College admission]
[Seattle Fire department calls]
[US mortality rates]

Git Collaboration


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Please don't....

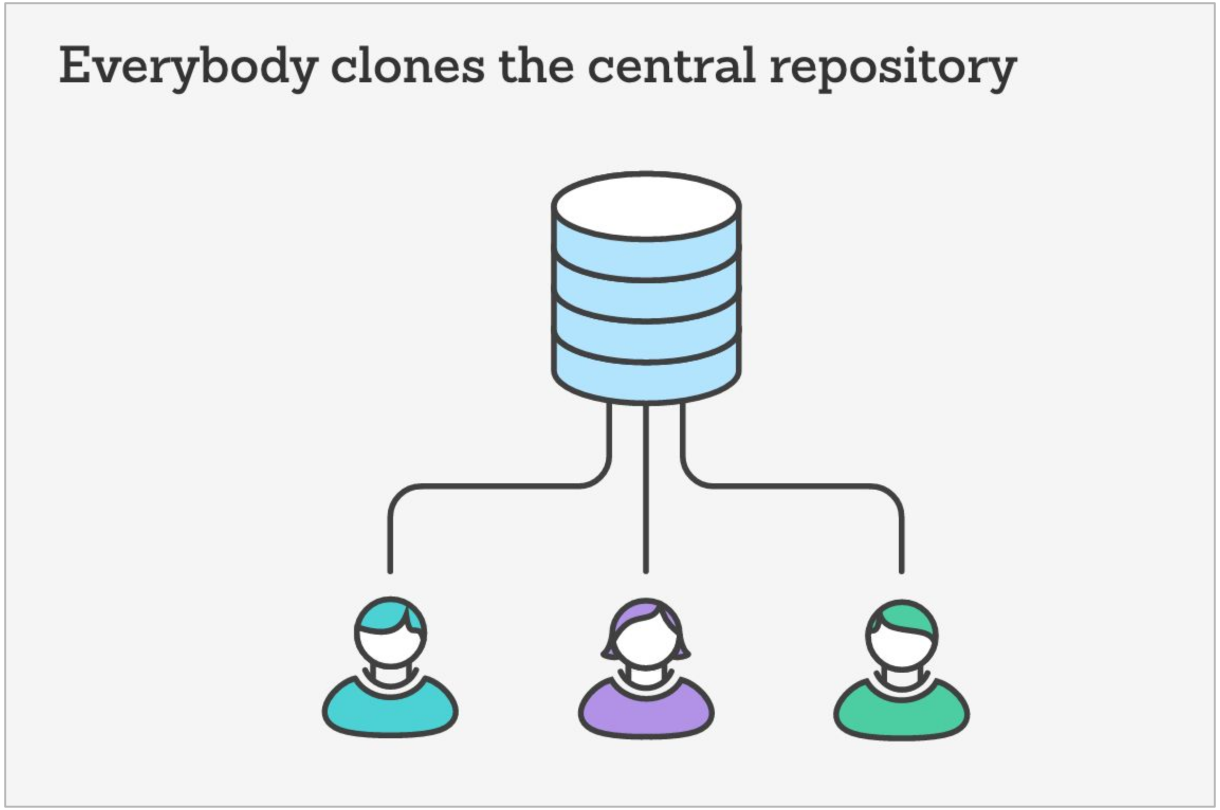
Challenges of Collaboration

Work on the same file at the same time

Do work at different times

Code clarity

Breaking down your code into appropriate components



Centralized Workflow

module-13

Centralized Workflow

Only have **one remote repository**

Each person clones the same repository (don't fork, only clone)

The remote repository is the **single source of truth**

One of many workflows

Centralized Workflow: Set up

One person (owner) **creates the repository** on GitHub

The owner then **adds collaborators** on GitHub (gives read/write access)

Each person then **clones the repo** and works on their local machine

One of many workflows

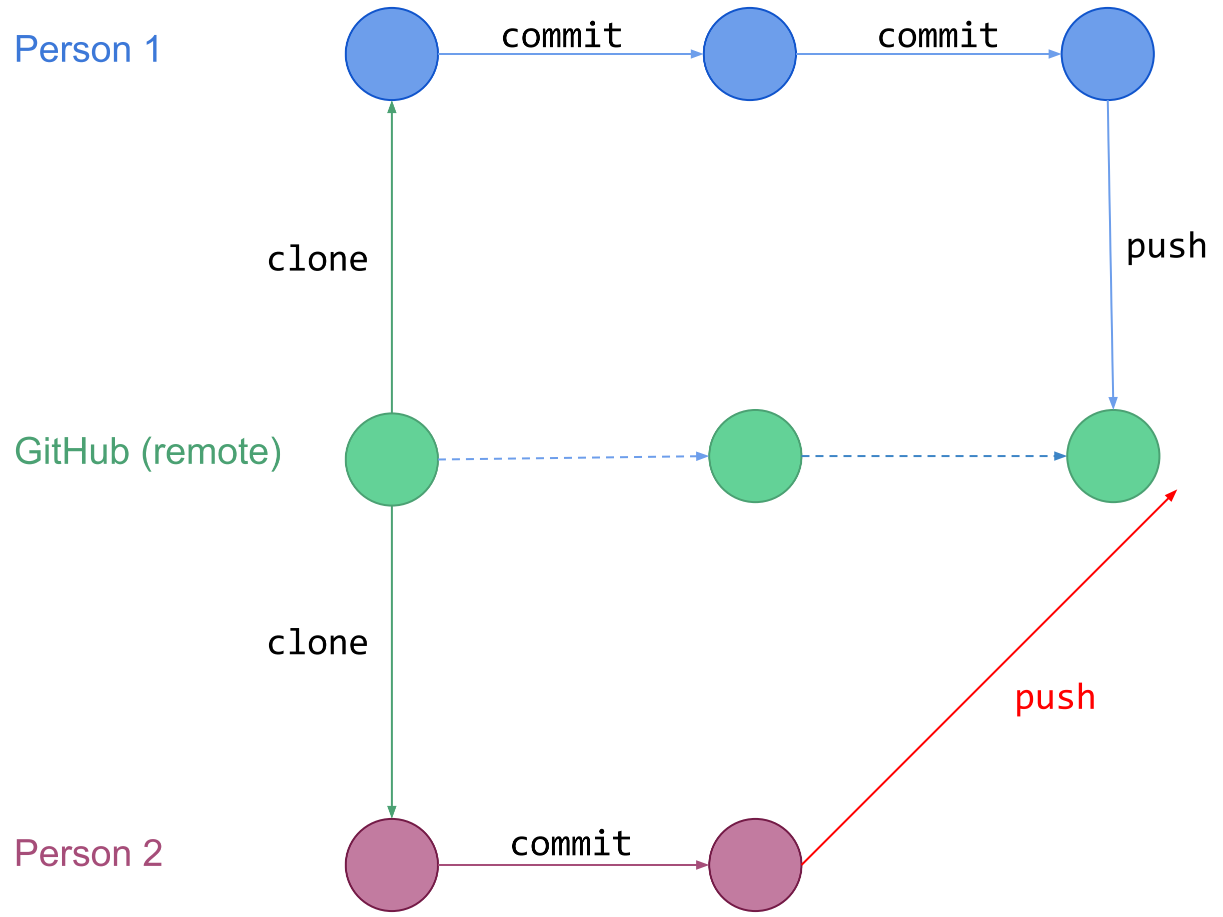
GitHub Issues

A great way to **assign and track** project tasks on GitHub

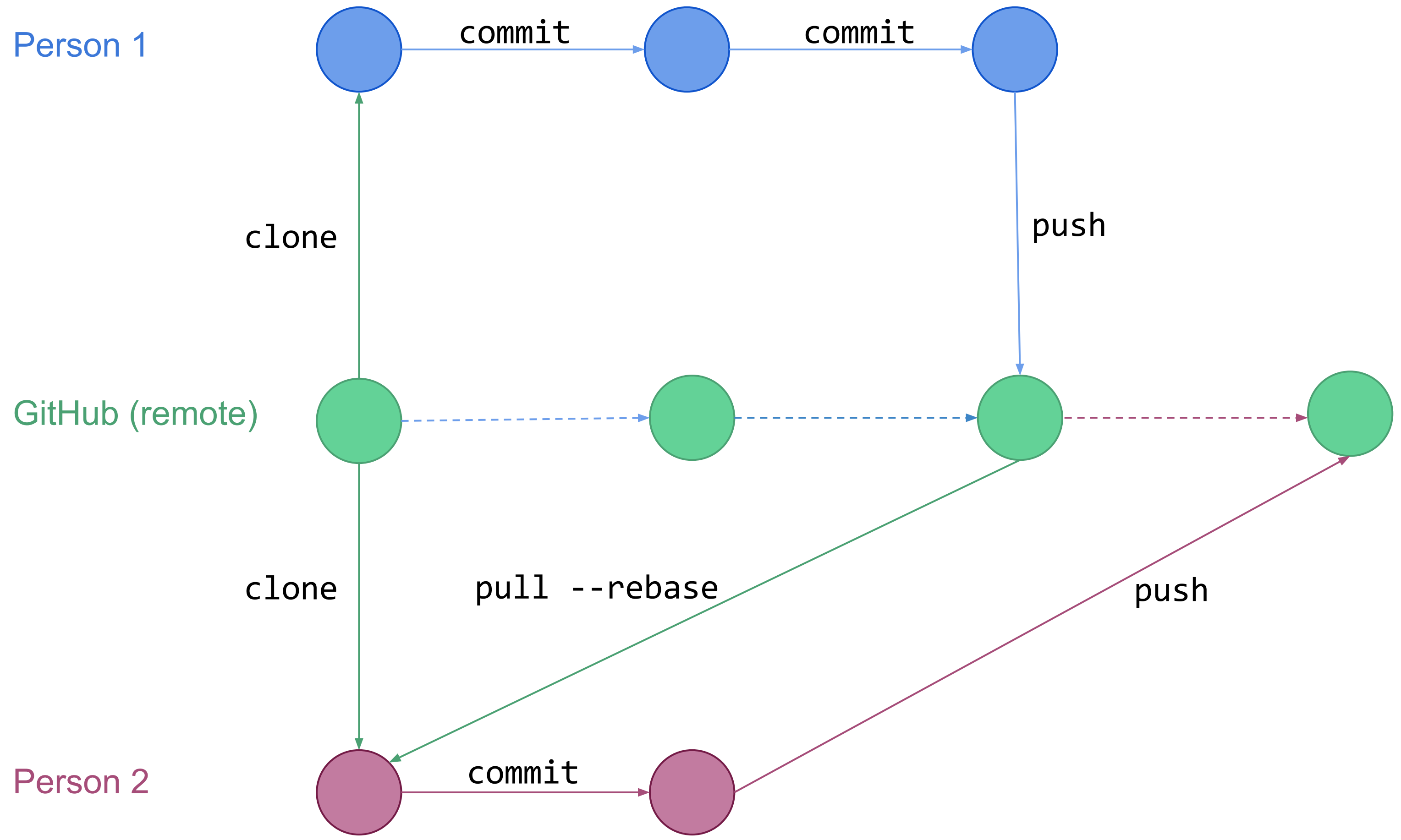
Helps you **plan** next steps in your project

Easily created on GitHub, then closed (via commit messages!)

Documentation



Multiple Contributions



Multiple Contributions

Rebasing

GitHub (thankfully) won't allow us to push if we aren't working on top of the **truth**

The **--rebase** option **pulls** down code, and integrates your changes with the changes on the remote

Achieves this by rewinding your code to a common ancestor, and the applying changes on top of it

```
# Person 2: add and commit local changes
git add .
git commit -m "Made changes"

# Attempt to push back to GitHub
git push origin master # Fails because not up to date!

# Rebase project from remote (origin) into master branch. I suggest starting most days like this...
git pull --rebase origin master

# Test your code to make sure it runs as expected!!!!!

# Push up changes to master
git push origin master
```

module 13 exercise-1

Conflict

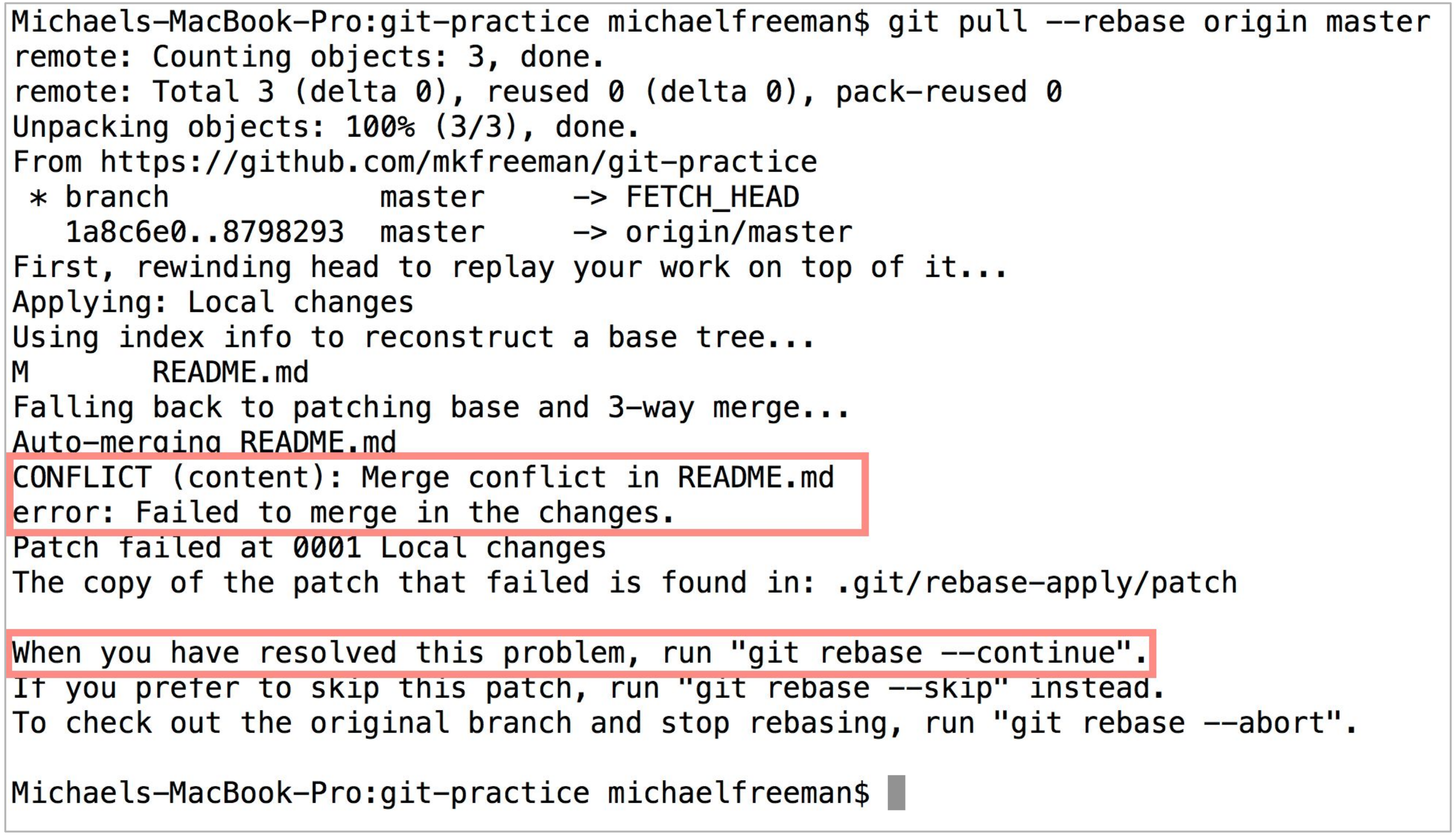
Resolving Conflict

Conflicts occur when different people edit the **same lines of code**

Just because there is not a conflict **does not mean** that your code works properly

If there is a conflict in a **rebase**, you'll have to open up your file and resolve it

This is a common part of the process and **should not be feared!**



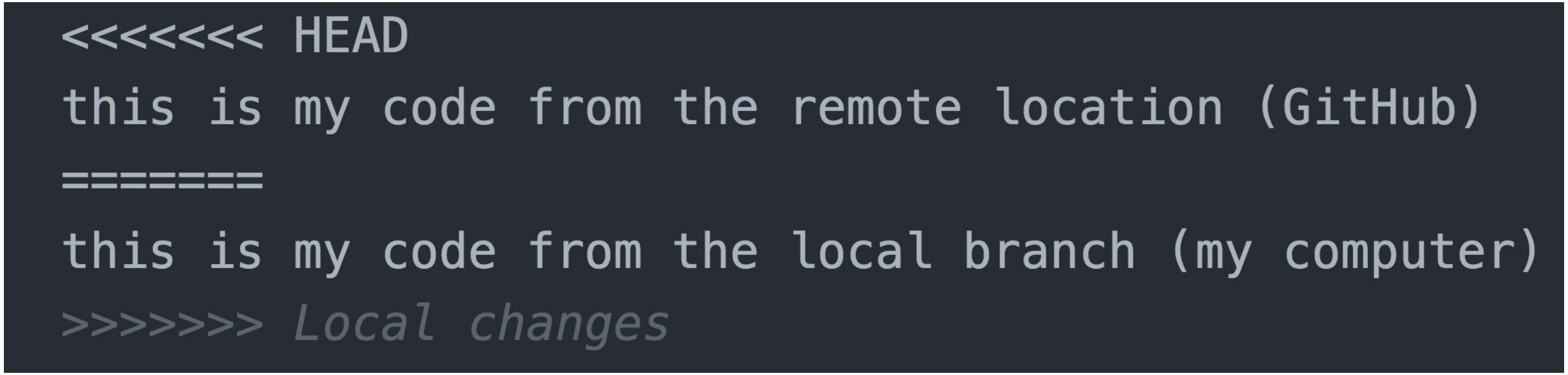
```
Michaels-MacBook-Pro:git-practice michaelfreeman$ git pull --rebase origin master
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/mkfreeman/git-practice
 * branch            master     -> FETCH_HEAD
   1a8c6e0..8798293  master     -> origin/master
First, rewinding head to replay your work on top of it...
Applying: Local changes
Using index info to reconstruct a base tree...
M       README.md
Falling back to patching base and 3-way merge...
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
error: Failed to merge in the changes.
Patch failed at 0001 Local changes
The copy of the patch that failed is found in: .git/rebase-apply/patch

When you have resolved this problem, run "git rebase --continue".
If you prefer to skip this patch, run "git rebase --skip" instead.
To check out the original branch and stop rebasing, run "git rebase --abort".

Michaels-MacBook-Pro:git-practice michaelfreeman$ 
```

Rebase conflict error

module-13



```
<<<<<<<< HEAD
this is my code from the remote location (GitHub)
========
this is my code from the local branch (my computer)
>>>>>>>> Local changes
```

This is what it does to your code!

```
# Add and commit local changes
git add .
git commit -m "Local changes"

# Attempt to push
git push origin master # doesn't let us push!

# Pull in and rebase remote changes
git pull --rebase origin master # oh no! conflict!

# Make manual changes in necessary files ----------

# Add changes that you've made to the rebase
git add .

# Continue (complete) the rebase process
git rebase --continue

# Push (integrated) changes up to GitHub
git push origin master
```

How to fix it

module-13

module 13 exercise-2

Branch Merging Conflicts

A **branch** in Git is a way of labeling a ***sequence of commits***. You can create labels (branches) for different commits, and effectively have ***different "lines" of development*** occurring in parallel and diverging from each other.

```
# Make and checkout new branch
git checkout -b my-branch

# Make some changes to your file, then add and commit
git add .
git commit -m "Made changes over here"

# Switch back to master branch
git checkout master

# Make some changes, then add and commit
git add .
git commit -m "Made changes to master"

# Merge in changes from my-branch branch
git merge my-branch

# Resolve the conflict MANUALLY in the file, then add and commit
git add .
git commit -m "Merged in changes fro my-branch branch"
```

Resolving merge conflicts

module-13

module 13 exercise-3

Upcoming...

By Thursday: Be confident with **module 13**

Due Tuesday, 11/15 (***before class***): project proposal, a7-collaboration

No lab next week

Data resources