A1: Git

Published ⋮

Let's *Git* To It (I am *so* sorry)

From the learning materials you have learned that Git is a tool used for Software Version Control (SVC). This is useful for single developer projects as well as larger teams. Being able to develop and test out new features without "corrupting" the main codebase is a really useful. It is also very helpful to be able to rollback code changes that caused things to break.

Motivation

The only way to really learn something is to do it. With that in mind, for this assignment you will be getting your hands dirty and actually implement the commands you learned this week. At the end you will have a repository (repo) that has a particular structure and history. You will need to follow the instructions exactly in order for your repo to match the required output.

Course Learning Outcome(s):

- **Apply** automated tools such as *make* and CVS in a realistic setting (CLO 1)

Module Learning Outcome(s):

- **Apply** Version Control System techniques

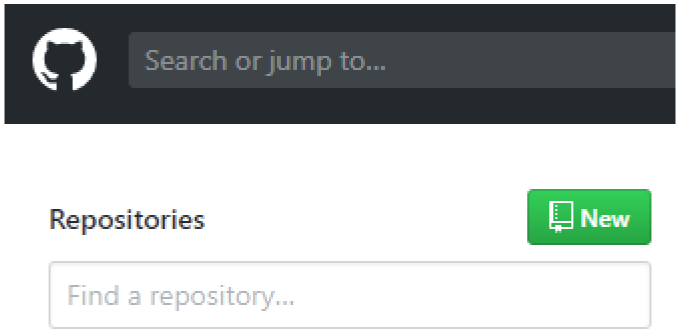
Description

Please Note: GitHub switched from using `master` to using `main` for the default branch while Git hasn't made the same changes. is means we will need to rename our default branch from `master` to `main` (covered in steps below)

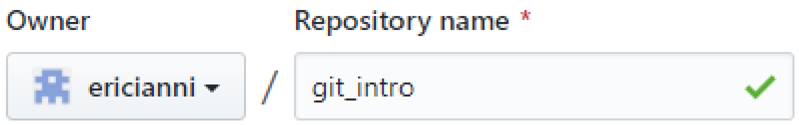
This is an all or nothing activity, but don't let that scare you. At the end of this document, you will find the expected structure. If you reach the end of the instructions and your repo does not match the expected output, just start over and pay close attention to the instructions.

Create a new GitHub repo

1. Log in to **github.com** **(https://www.github.com)**
2. On the left-hand side click the "New" repo button



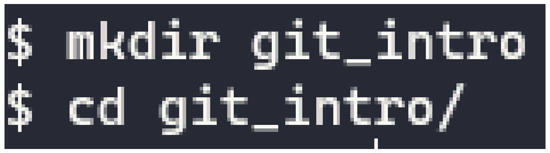
3. Name the repo `git_intro` and **uncheck** create README.md



4. Finally, click "create repository"

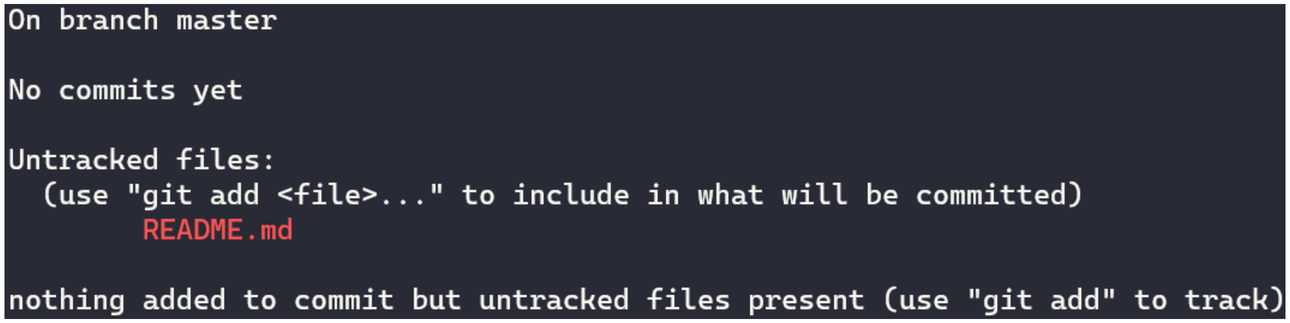
Create a local repo on Flip (or your local machine)

4. Log into Flip and create a new directory called `git_intro` and navigate into it



```
$ mkdir git_intro
$ cd git_intro/
```

5. Create a new Git repo by typing `git init`.
6. In this folder create a file called `README.md` and put your name into it. You can do this from the command line using `echo "Eric Ianni" > README.md`
7. Check the status of your repo by typing `git status`. You will see that `README.md` is *untracked* (You will also see that the branch is called `master` which we can only change after we commit)



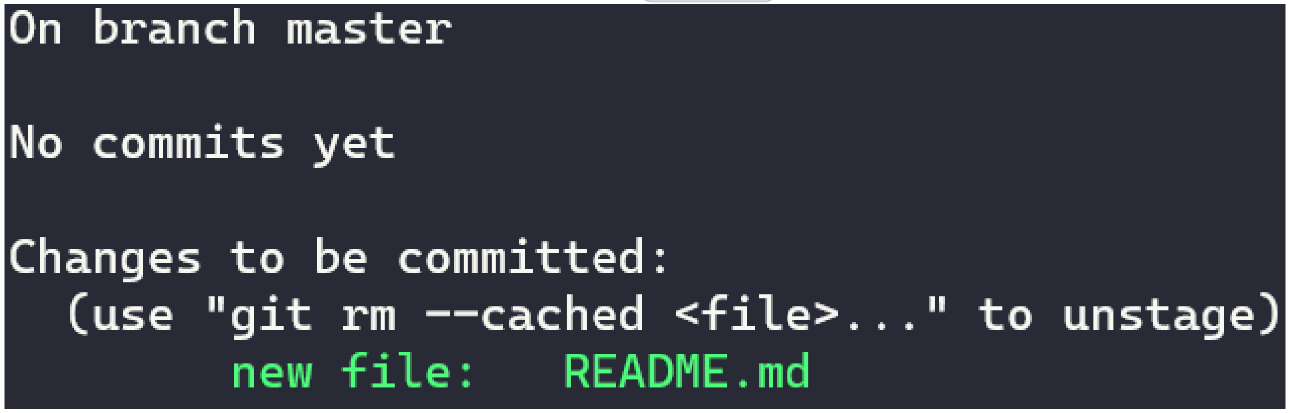
```
On branch master

No commits yet

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

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

8. In order to track, and eventually commit, `README.md` we need to *add* it to the repo. You can do this using `git add README.md`
9. Check the status again and you will see that `README.md` is now *tracked*, but you have no commits



```
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
```

10. Let's commit our changes by using `git commit -m "first commit"`. The `-m` tag allows us to add the commit comment inline. You may need to set some Git config variables for your email and name. You can set global variables using the following:

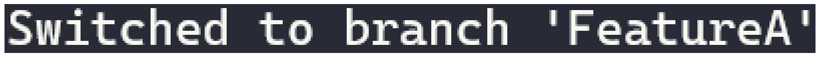
```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

    Be aware that if manage multiple GitHub accounts from the same computer, you may run into issues with setting global variables, but that is beyond the scope of this class. If you find yourself in this situation, reach out and I can point you in the correct direction.

11. If you are running this command after October 1st, 2020, you will need to rename `master` to `main`. You can do that with `git branch -m master main`. Please note, you can only do this once you have at least one commit to `master`

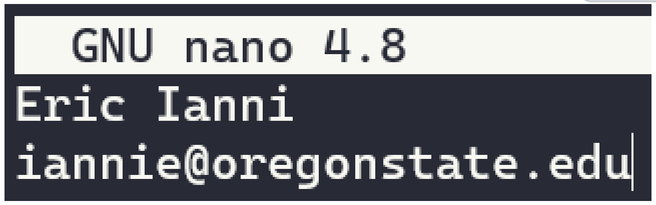
eating a new branch

11. Now it is time to branch out. Create a new branch with `git branch featureA` and check it out with `git checkout featureA`



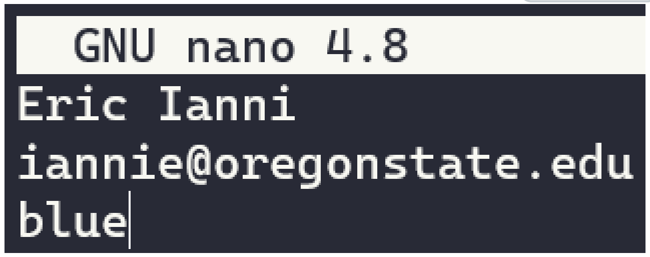
```
Switched to branch 'FeatureA'
```

12. Using the editor of your choice, open up `README.md` and on a new line add your email address



```
  GNU nano 4.8
Eric Ianni
iannie@oregonstate.edu
```

13. Add these changes to the `featureA` branch and commit like we did before: `git add README.md` followed by `git commit -m "added email"`
14. Let's make more changes! Open up `README.md` again and on a new line add your favorite color

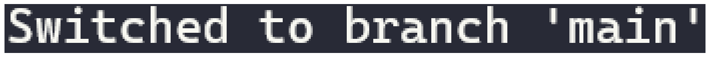


```
  GNU nano 4.8
Eric Ianni
iannie@oregonstate.edu
blue
```

15. Add these changes to the current branch and commit with the comment `added color`

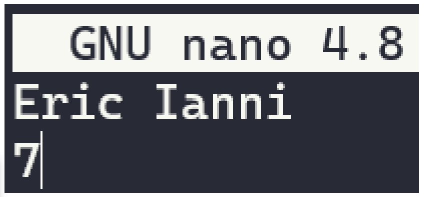
Back to main

16. To fully illustrate how branches work, checkout the master branch again: `git checkout main`



```
Switched to branch 'main'
```

17. Now when you open `README.md` what do you see? The changes you made on `featureA` are not reflected on `main`. Go ahead and add your favorite number

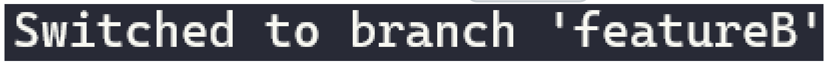


```
  GNU nano 4.8
Eric Ianni
7
```

18. Again, add these changes to the current branch with the comment `added number`

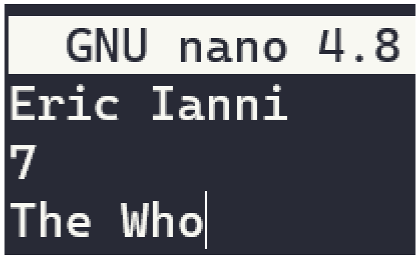
Creating another branch: Part Deux

19. Now create a third branch called `featureB` and check it out



```
Switched to branch 'featureB'
```

20. As we are wont to do, open up `README.md` and on a new line add the name of your favorite band

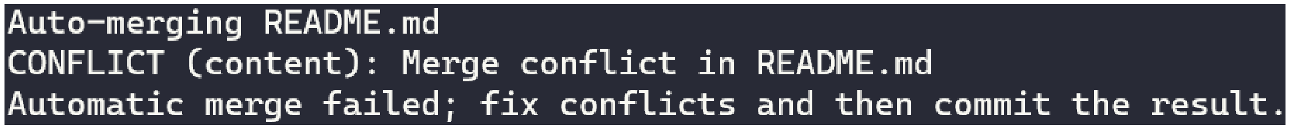


```
  GNU nano 4.8
Eric Ianni
7
The Who
```

21. Add these changes to the current branch and commit with the comment `added band`

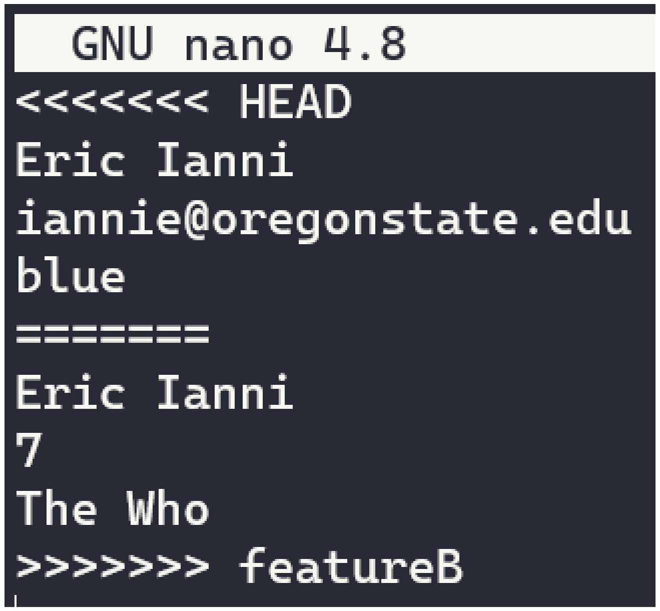
Time to merge

22. Before we *merge* let's introduce some *conflict*. Checkout the `main` branch again
23. Open up `README.md` and *remove* your favorite number
24. Add these changes to the current branch with the comment `removed number`
25. Checkout the `featureA` branch
26. Now attempt to merge the current branch with the `featureB` branch using `git merge featureB`
27. Oh no! The merge failed due to conflicts at we must resolve



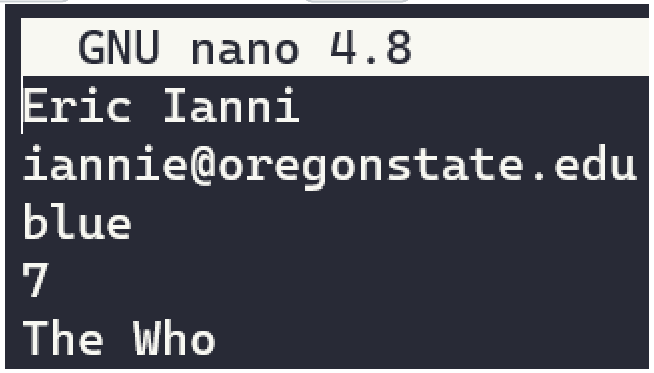
```
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

28. In order to resolve these conflicts we need to, once again, open up `README.md` and examine them. Below is how this looks in Nano on my local machiner



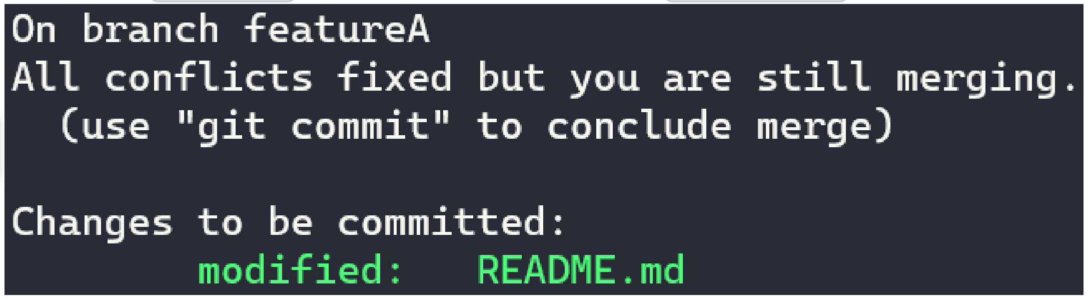
```
  GNU nano 4.8
<<<<<<< HEAD
Eric Ianni
iannie@oregonstate.edu
blue
=======
Eric Ianni
7
The Who
>>>>>>> featureB
```

29. The contents of `README.md` on the current branch (indicated by `HEAD`) are above the `=======` and the branch being merged below. Let's resolve this by keeping all the differences by removing the lines: `<<<<<<< HEAD`, `=======`, `>>>>>>> featureB`, and `Eric Ianni` from below the `=======`



```
  GNU nano 4.8
Eric Ianni
iannie@oregonstate.edu
blue
7
The Who
```

30. If you run `git status` you will see that you are `still merging` and need to commit to finish the process



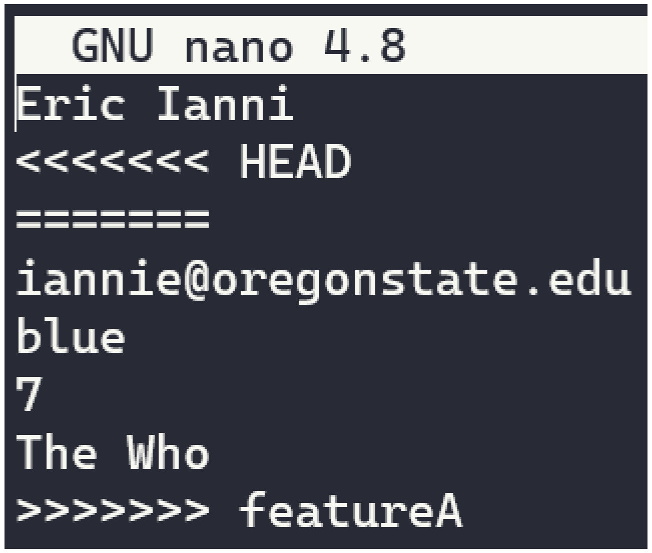
```
On branch featureA
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        modified:   README.md
```

31. If your status doesn't show `README.md` as already green, add it before proceeding. Go ahead and complete the merge by committing with the message `merged featureA and featureB`

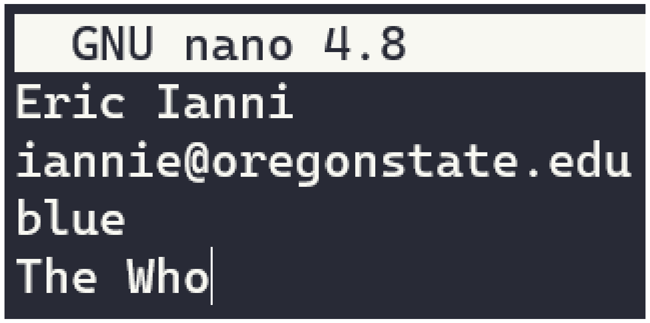
Bringing it all back to main

32. Let's bring all the branches together. Check out the `main` branch
33. Now attempt to merge `featureA` into the current branch like we did before
34. Darn! The merge failed again because of conflicts. Open up `README.md` and take a look



```
  GNU nano 4.8
Eric Ianni
<<<<<<< HEAD
=======
iannie@oregonstate.edu
blue
7
The Who
>>>>>>> featureA
```

35. This time let's keep everything *but* the favorite number



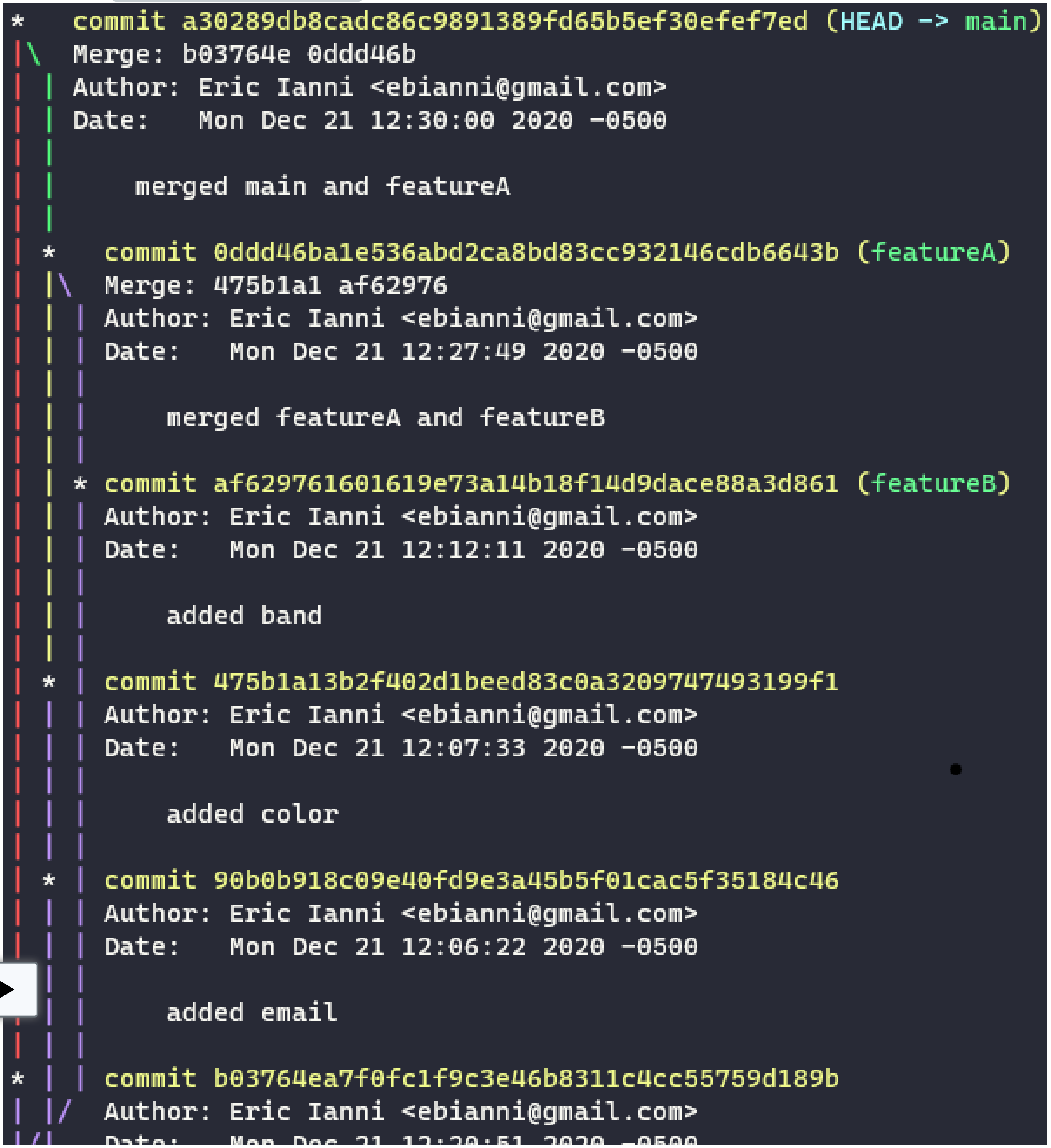
```
  GNU nano 4.8
Eric Ianni
iannie@oregonstate.edu
blue
The Who
```

36. Complete the merge by committing with the message `merged main and featureA`

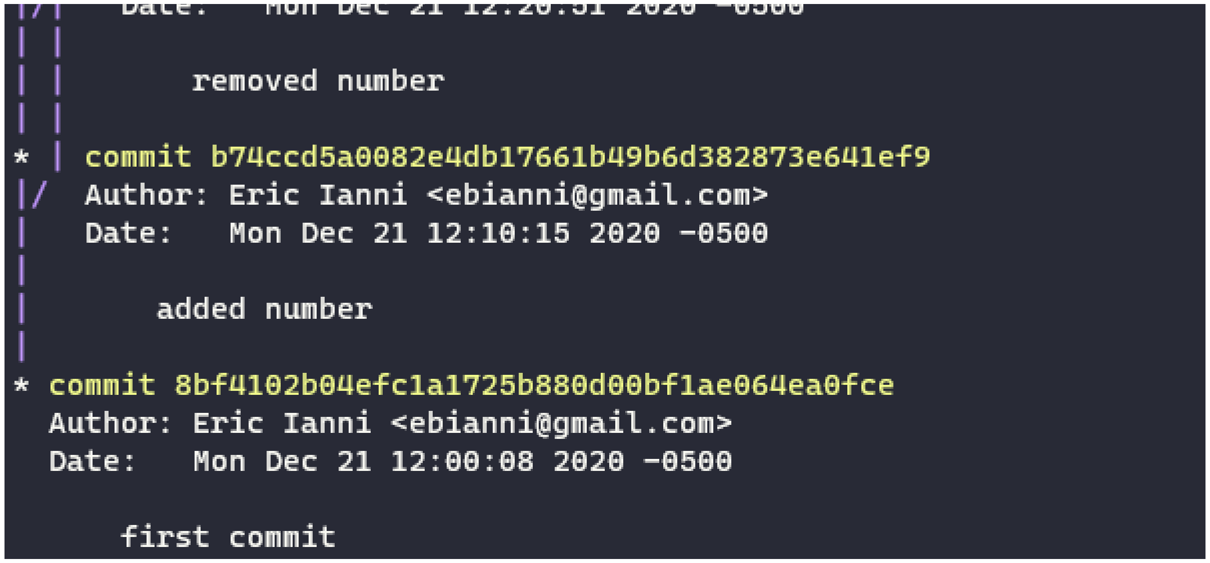
Where does this leave us?

During this assignment, we have created two additional branches, each with unique changes. We have also merged all these branches back together. It can be hard to mentally picture all these changes. Luckily there is a git command to help!

37. Run `git log --graph --all` to generate a tree of your commits and merges. Yours should look like this:



```
*   commit a30289db8cadc86c9891389fd65b5ef30efef7ed (HEAD -> main)
|\  Merge: b03764e 0ddd46b
| | Author: Eric Ianni <ebianni@gmail.com>
| | Date:   Mon Dec 21 12:30:00 2020 -0500
| |
| |     merged main and featureA
| |
| *   commit 0ddd46ba1e536abd2ca8bd83cc932146cdb6643b (featureA)
| |\  Merge: 475b1a1 af62976
| | | Author: Eric Ianni <ebianni@gmail.com>
| | | Date:   Mon Dec 21 12:27:49 2020 -0500
| | |
| | |     merged featureA and featureB
| | |
| | * commit af629761601619e73a14b18f14d9dace88a3d861 (featureB)
| | | Author: Eric Ianni <ebianni@gmail.com>
| | | Date:   Mon Dec 21 12:12:11 2020 -0500
| | |
| | |     added band
| | |
| * | commit 475b1a13b2f402d1beed83c0a3209747493199f1
| | | Author: Eric Ianni <ebianni@gmail.com>
| | | Date:   Mon Dec 21 12:07:33 2020 -0500
| | |
| | |     added color
| | |
| * | commit 90b0b918c09e40fd9e3a45b5f01cac5f35184c46
| | | Author: Eric Ianni <ebianni@gmail.com>
| | | Date:   Mon Dec 21 12:06:22 2020 -0500
| | |
| | |     added email
| | |
* | | commit b03764ea7f0fc1f9c3e46b8311c4cc55759d189b
| |/  Author: Eric Ianni <ebianni@gmail.com>
|/|   Date:   Mon Dec 21 12:20:51 2020 -0500
```

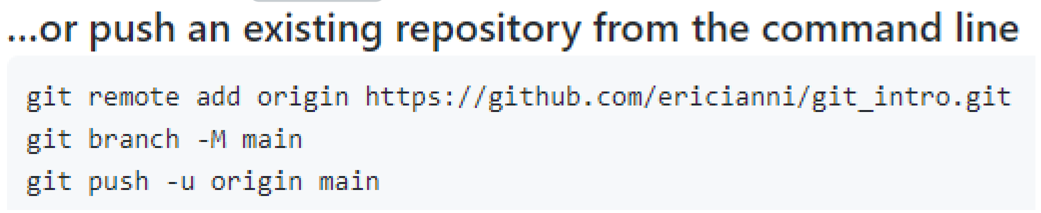


```
| |
| |       removed number
| |
* | commit b74ccd5a0082e4db17661b49b6d382873e641ef9
|/  Author: Eric Ianni <ebianni@gmail.com>
|   Date:   Mon Dec 21 12:10:15 2020 -0500
|
|       added number
|
* commit 8bf4102b04efc1a1725b880d00bf1ae064ea0fce
  Author: Eric Ianni <ebianni@gmail.com>
  Date:   Mon Dec 21 12:00:08 2020 -0500

      first commit
```

Pushing it to GitHub

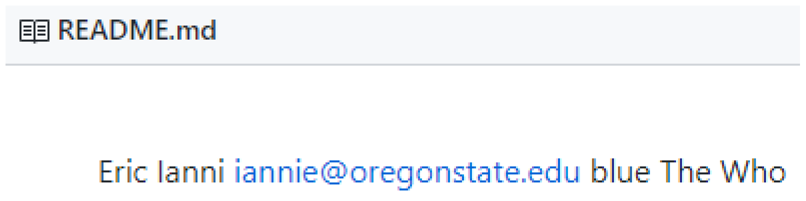
38. Back at your new `git_intro` repo on GitHub you should see a section detailing how to push an existing repo



...or push an existing repository from the command line

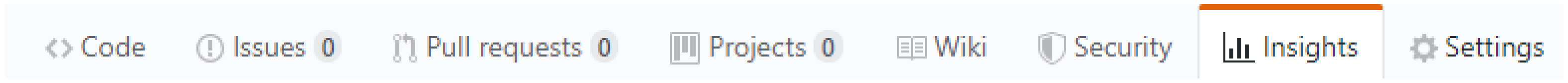
```
git remote add origin https://github.com/ericianni/git_intro.git
git branch -M main
git push -u origin main
```

39. On Flip, use the URL you see in your `git_intro` repo to add a remote: `git remote add origin https://github.com/ericianni/git_intro.git (https://github.com/ericianni/git_intro.git)` (make sure you have HTTPS selected, not SSH)
40. Please note, that we have already run `git branch -M main`, so we do not need to do so again
41. *If* we just wanted to push the current branch we would use `git push -u origin main`, but we want to push *all* of our branches! Therefore use `git push --mirror origin`. Please note, GitHub does not allow for passwords to be used to authenticate yourself on the command line. You will need to use a Personal Access Token as described in **Exploration: Set Up Git and Conda**
42. Go back to your GitHub `git_intro` repo and you will see your updated `README.md`. It will all be on one line because GitHub uses something called *markdown* which has a different syntax than you are used to

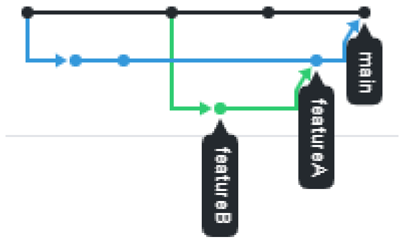

README.md

Eric Ianni iannie@oregonstate.edu blue The Who

43. Just like on the command line, GitHub has a way to visually inspect your branches and commits. Click the `Insights` tab



44. To see the tree click `Network` on the left



What To Turn In

For this assignment, you need to provide us with a link to your *public* `git_intro` GitHub repo that matches the structure shown above. It needs to be public to ensure we can view it.

Failure to submit a working link will resort in a 0 on the assignment. If we can't access your repo link for any reason (e.g. it is private), you will receive a 10% penalty even if you supply a working link later. Please double-check the link you provide by attempting to open it in Incognito Mode.

Points	10
Submitting	a website url

Due	For	Available from	Until

Due	For	Available from	Until
Jan 10	Everyone else	-	-
Jan 14	1 student	-	-

Activity 1 Rubric

Criteria	Ratings		Pts
GitHub Repo Matches The commit history and network graph is the same as shown in the assignment All branches are pushed (see 'mirror' in the steps above)	**10 pts** **Full Marks**	**0 pts** **No Marks**	10 pts
			Total Points: 10

▶