Team Git Practice

Need a tutorial on Git: branching? \https://www.freecodecamp.org/news/git-and-github-for-beginners/

Let's simulate using Git in a team environment to help better prepare for development in a professional setting, or in the Capstone course.

Preparation

You'll be split into teams of 3-4. If You're in a group of 3, then the Release Manager will also play the role of Developer 3. These steps should be done in order.

You'll need to share GitHub names in order to collaborate. Send your user name to the Release Manager.

- Release Manager - Responsible for managing the remote repository, merging branches
- Developer 1
- Developer 2
- Developer 3

The person who should be following a series of steps will be marked like `DEVELOPER 1`

Exercise

RELEASE MANAGER

1. Create a public repository in your GitHub account, name the repository GitLab
2. Add the 2-3 additional developers in your team as Collaborators, also include your professor as a Collaborator. To invite Collaborators, click on the Settings tab in your repository.

EVERYONE

1. Once you've accepted the invitation, find the GitLab repository. Click on the green Code button to find the ***url*** of the new repository.
2. Clone the newly created repository to your computer (you can use any location for this exercise, your Desktop for example).
3. git clone ***url.***
4. In the terminal window, navigate to the working directory you just cloned, GitLab.

DEVELOPER 1

1. Create a new branch called `foundation` and switch to it; `git checkout -b foundation`.
2. Create a file named `index.html`, save to the GitLab folder.
3. Add the base HTML structure into the page, don't worry about including anything else
4. Stage the file for your next commit; `git add index.html`
5. Check the status of your staged file; `git status`
6. Commit the file, with a message; `git commit -m "<your last name>:initial index.html file"`
7. **Stop and Think**: Do the other developers have access to the file you created yet?
8. Push the branch to the remote repository; `git push -u origin foundation`.
9. Go to remote repository and create a pull request comparing the foundation branch to main.

RELEASE MANAGER

1. Go to GitHub and view the repository you created for your team
2. Click on "Pull requests"
3. Review the changes being proposed by Developer 1
4. If they look good, approve the changes by clicking on Merge pull request. Confirm merge.

DEVELOPER 2

1. Pull down the latest changes from the remote repository; `git fetch`
2. Create a new branch named `quotes` and switch to it
3. The index.html page is in need of some content. Add an `<h1>` element with a title for the page
4. Additionally, add a `<blockquote>` element to the page with your favorite quote, be sure to include the author
5. Go ahead and create a file named `style.css`, but don't worry about adding anything to it yet
6. Stage your changes for the next commit
7. Commit your changes, adding a pertinent message, "`<your last name>: … "`
8. Push the branch to the remote repository and create a pull request.

RELEASE MANAGER

1. Go to GitHub and view the repository you created for your team
2. On the right hand side click on "Pull requests"
3. Review the changes being proposed by Developer 2
4. If they look good, approve the changes by merging the new `quotes` branch into `master`

DEVELOPER 3

1. Pull down the latest changes from the remote repository
2. Create a new branch named `styles` and switch to it

3. Add a couple of very basic styles in `style.css` (e.g. add a background color, style the quote). Add a link to style.css in the `index.html` file.
4. Stage your changes for the next commit
5. Commit your changes, adding a pertinent message, " `<your last name>: … "`
6. You think that the author attributed to the quote is incorrect. You believe it's supposed to be **Justin Bieber**. Change the author to reflect this
7. Stage your changes for the next commit
8. Commit your changes, adding a pertinent message, "`<your last name>: … "`
9. Push the branch to the remote repository and create a pull request

RELEASE MANAGER

1. Go to GitHub and view the repository you created for your team
2. On the right hand side click on "Pull requests"
3. Review the changes being proposed by Developer 3
4. You don't notice the error with the author, so just approve the changes by merging the new `styles` branch into `master`

DEVELOPER 1

1. Your biggest client notices that the author of the quote is completely incorrect. Developer 3 is on vacation, and you're tasked to fix this
2. Pull down the latest changes from the remote repository, make sure you are in the correct location, you should be in the main area, not your branch. To see your location; `git status` to see your branches; `git branch -a`
3. To switch to the main branch; `git checkout main`
4. Create a new branch named `hotfix1` and switch to it
5. Correct the author
6. Stage your changes for the next commit
7. Commit your changes, adding a pertinent message"`<your last name>: … "`
8. **DO NOT** push your changes to the remote repository yet

DEVELOPER 2

1. Pull down the latest changes from the remote repository
2. Create a new branch named `authorupdate` and switch to it
3. Your manager asked you to make the author's name all uppercase, so go ahead and do that.
4. Stage your changes for the next commit
5. **DO NOT** commit your changes or push them to the remote repository yet

DEVELOPER 1

1. Push the `hotfix1` branch to the remote repository and create a pull request

RELEASE MANAGER

1. Go to GitHub and view the repository you created for your team
2. On the right hand side click on "Pull requests"
3. Review the changes being proposed by Developer 1
4. Approve the changes by merging the new `hotfix1` branch into `master`

DEVELOPER 2

1. Switch to the `master` branch
2. Pull down the latest changes from the remote repository
3. You noticed some files have changed, and you want the new branch you're working on to also have these changes
4. Switch to the existing `authorupdate` branch that you're still working on
5. Run `git merge master` to get those updates
6. You should have a conflict now with the new changes clashing with yours. Resolve these conflicts
7. Stage your changes for the next commit
8. Commit your changes, adding a pertinent message, "`<your last name>: … `"
9. Push the branch to the remote repository and create a pull request

RELEASE MANAGER

1. Go to GitHub and view the repository you created for your team
2. On the right hand side click on "Pull requests"
3. Review the changes being proposed by Developer 2
4. Approve the changes by merging the new `authorupdate` branch into `master`

DEVELOPER 3

1. Pull down the latest changes from the remote repository
2. Create a new branch named `hotfix2` and switch to it
3. You're absolutely certain the author's name is supposed to be **Justin Bieber**. Change it back to that again
4. Stage your changes for the next commit
5. Commit your changes, adding a pertinent message, "`<your last name>: … `"
6. Push the branch to the remote repository and create a pull request

RELEASE MANAGER

1. Go to GitHub and view the repository you created for your team
2. On the right hand side click on "Pull requests"
3. Review the changes being proposed by Developer 3
4. This time, you learned your lesson and instead of merging you leave a comment about how this is incorrect and just close the request without merging.
5. Ask Developer 3 to stop doing this and tell them there's more to life than Justin Bieber

Exercise modified, source: https://gist.github.com/mdang/9ef6df246e23a795936b