Git and using GitHub

Unless you don’t like being able to work with other people and not seeing the changes you made in the past

FIRST THING FIRST!

- Git ≠ GitHub

- Git is **Form** of Version Control Software
  - AKA source control
  - Many other forms
- GitHub is a place to host your Git Repositories online
  - Other websites offer this too like BitBucket

Ways to use Git

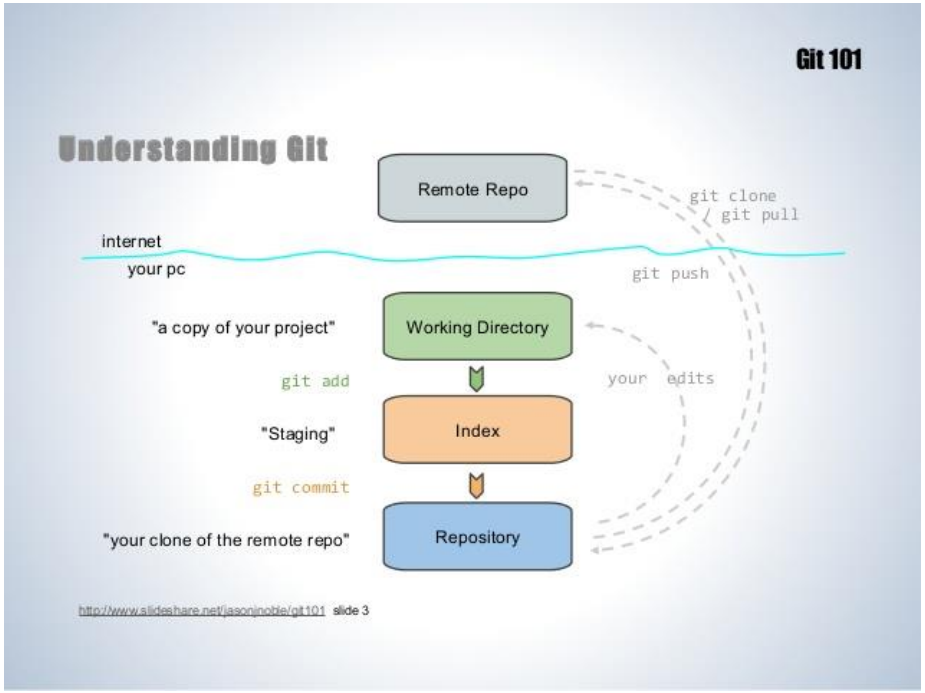
- There are pretty much two main ways to use Git
  - Command Lines
    - **Pros**: You are really forced to understand Git
    - **Cons**: You are really forced to understand Git
  - GUI Tools (GitHub Desktop, Git Extensions, etc)
    - **Pros**: Way easier to use and manage code
    - **Cons**: This is how you look to some people

Gotta Get Git

- Command Line
  - https://git-scm.com/downloads
    - Git Bash is just a Windows side application that opens a command prompt with Git
- GitHub Desktop GUI (what I recommend)
  - https://desktop.github.com/

Git 101- What happens to code

- Repository (the code) is saved on local computer
- Repositories can then be **Pushed** to a remote server where other people can push too also



Git 101 – It Remembers …so you don't have too

- "I'll just save the code to a flash drive and give it to you"
  - If you say this again instead of using Git you will wake up as a Political Science Major
- Git saves all past **Committed** saves in a .git folder in the repository
- New people can go back to ANY old **Commit** made during life of repository

Git 101 – Workflow

- Obtain a repository
  - If you have the repository already, **Fetch** the latest version to sync up with it
- Make your edits
- **Stage** your changes
- **Commit** your work
- **Push** to remote location so combine
- Wait for other people to pull their weight and add code to the repo, then **Pull** it

Git 101 - Branches

- Branches let you work on the code in your own crazy direction and **Merge** it back later
- Example: Make a “New-Feature” branch and when it is ready, **Merge** back to the Master Branch

Git 101 – The “Magic Moment”

- “Wait, it updated the folder for me!?!?”
  - You will say this at least once
- The “Magical” part of using Git is whenever you **Pull** from the latest commits, change **Branches**, or **Revert** to an old **Commit** in the Repo the folders on your local machines will reflect the changes
- If you had a file called “Hello.c” deleted a few commits ago and you go back the file will be back in your folder on your machine…Magic!

GitHub GUI – Make new repo

- Can either create online and bring it to local computer
- Start new repository and push to GitHub when ready (If using a private repo, why wait)
  - HUGE SIDE NOTE: Go here if you haven't before https://education.github.com/pack/offers and get your free private repositories
- Can take a current set of code and make it into a Git repo
  - Will have no history prior to initialization of git

How to use GUI

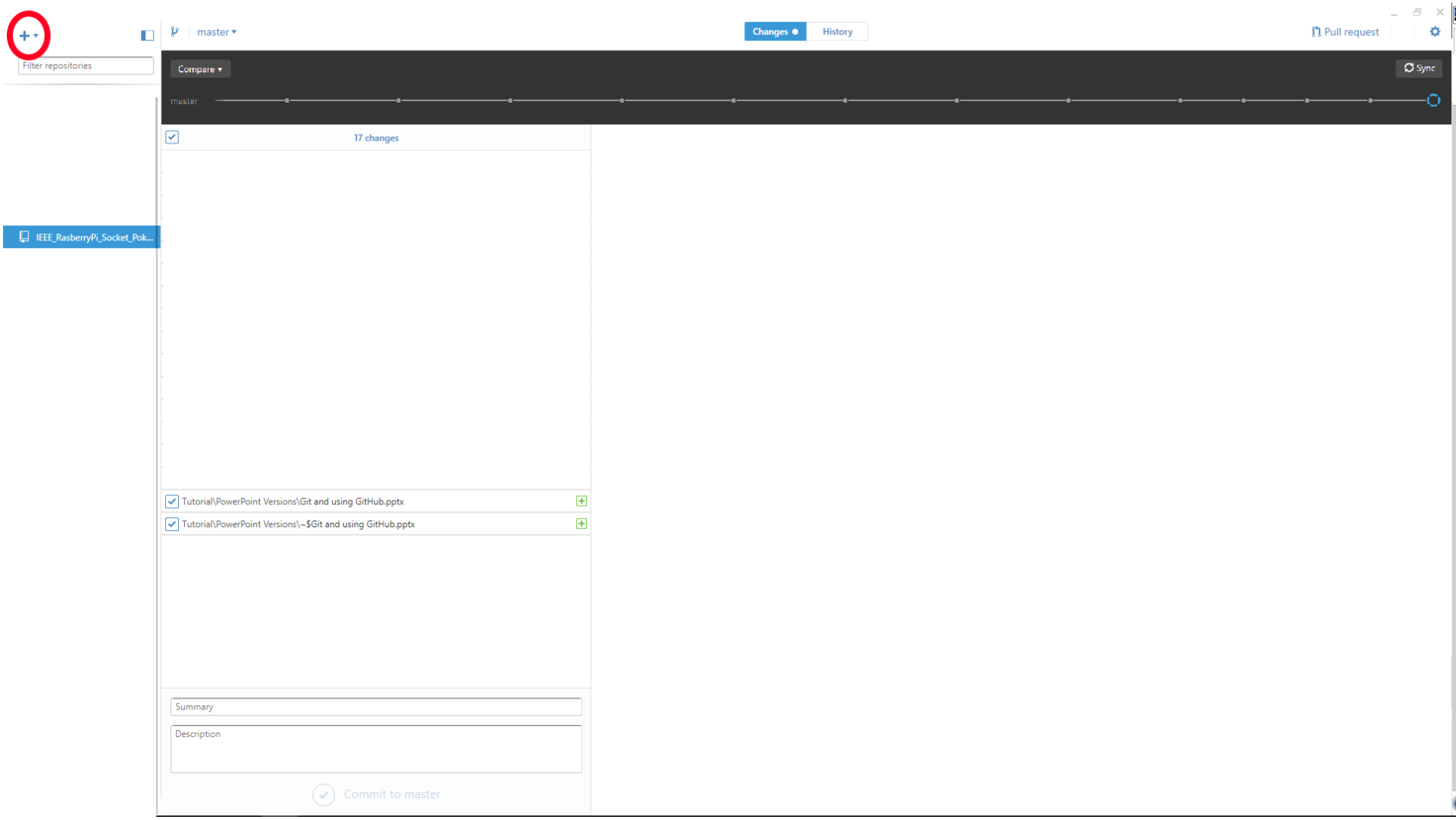
- If you want to learn more go here: https://guides.github.com/

- If you want to get this repo go here
- https://github.com/sjfricke/IEEE_RaspberryPi_Socket_Pokemon

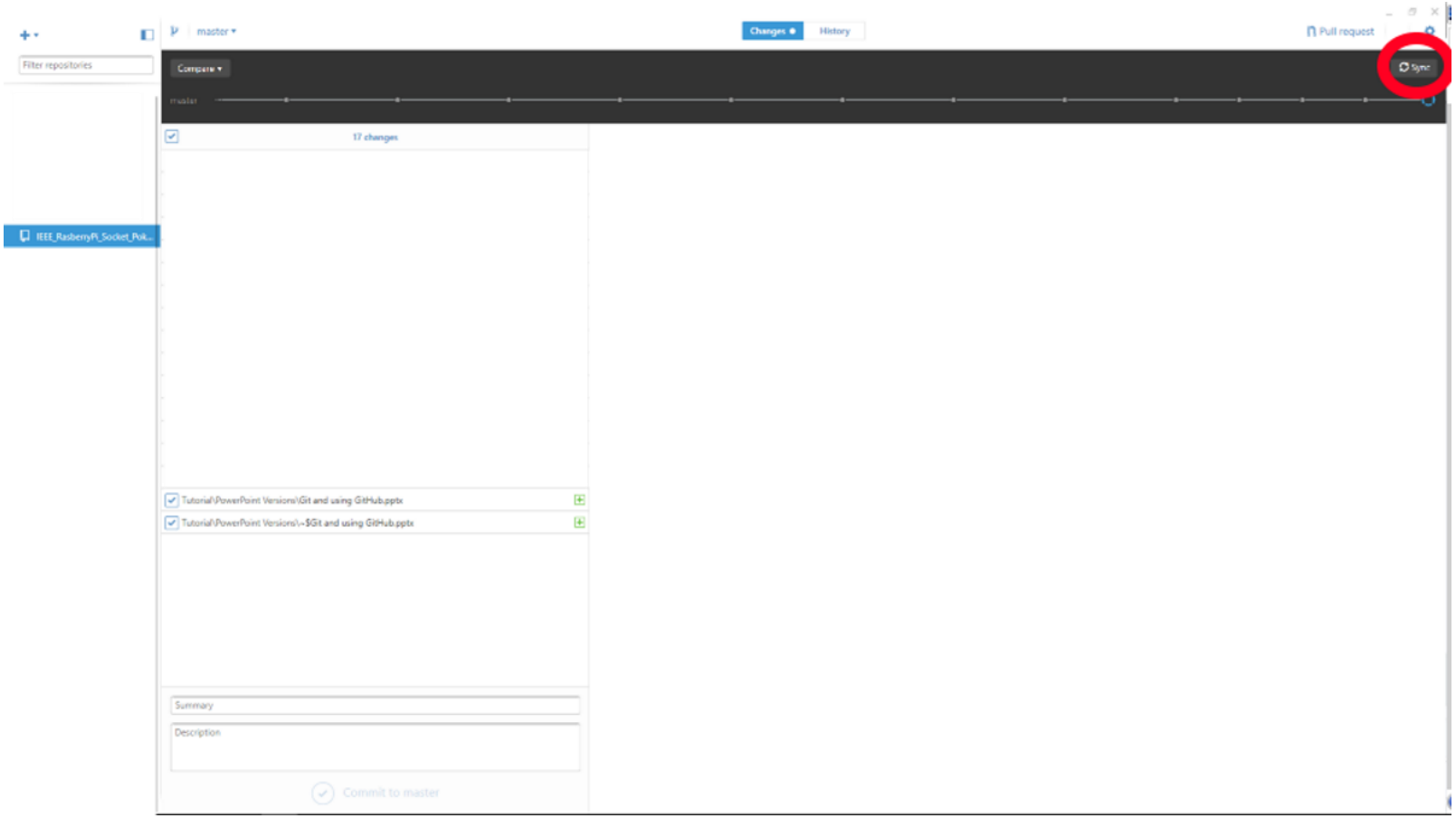
- Click the “Clone or Download” button and Clone it to your desktop (will open Desktop GUI for you)

GitHub Permissions

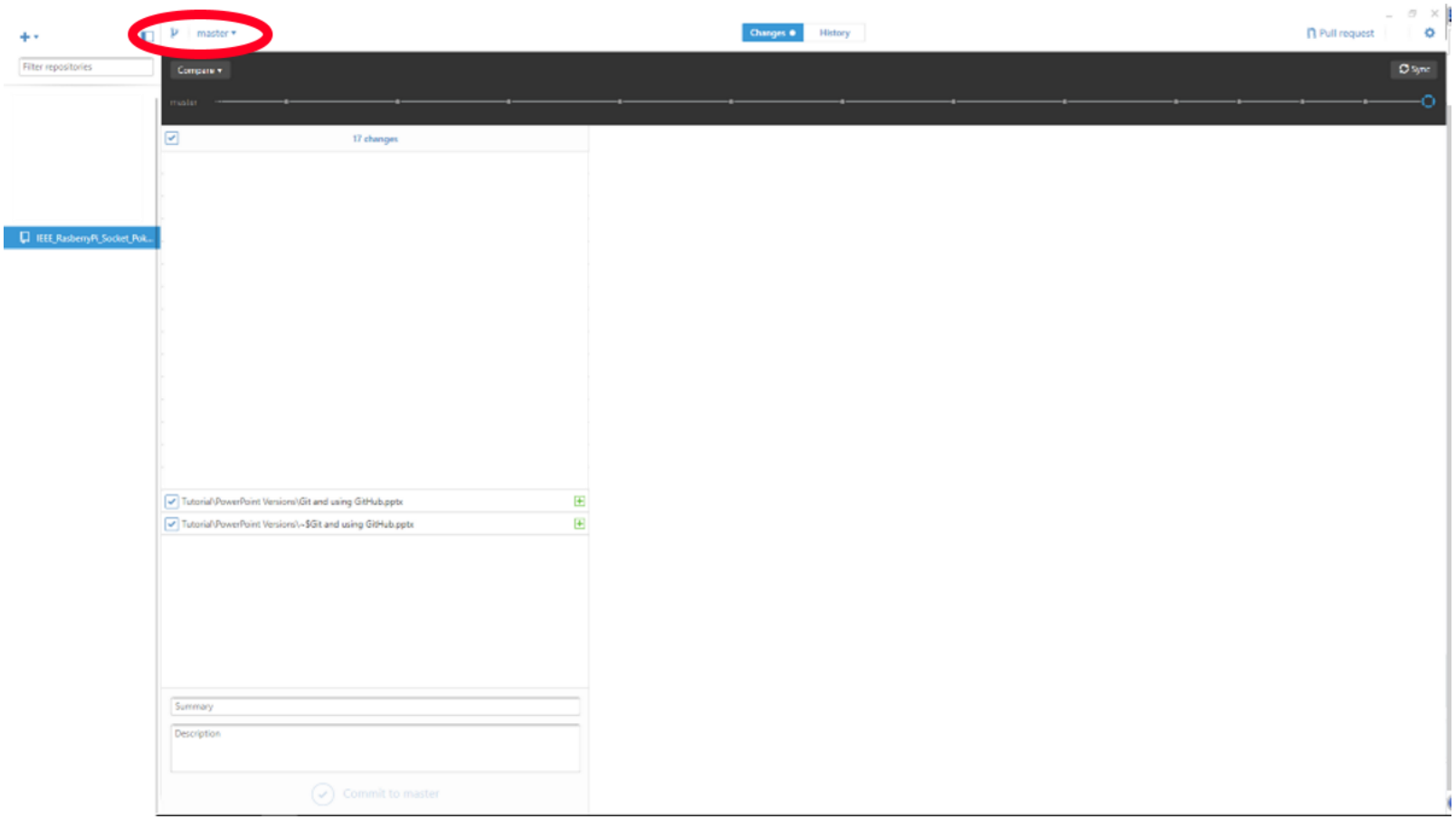
- For a project you can always just download the code and do what you want.
- If you want to make changes, either **Clone** or **Fork** the Repository
  - You can then send a **Pull Request** that will let someone in charge of Repo check your changes and **Merge** it
- If you set someone as a **Collaborator** they can **Push** code without having to submit a **Pull Request**



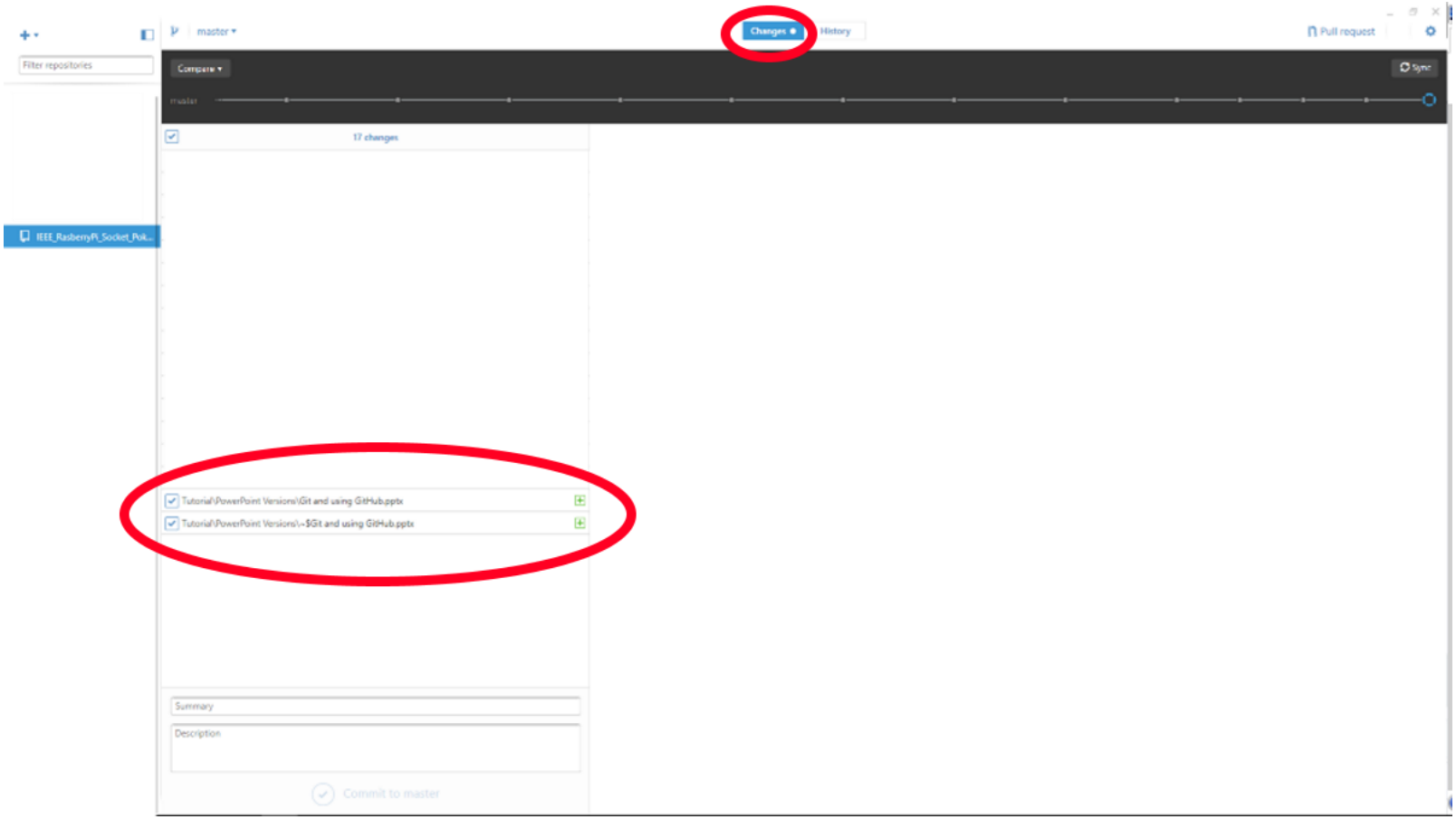
- Click here to add the Repo



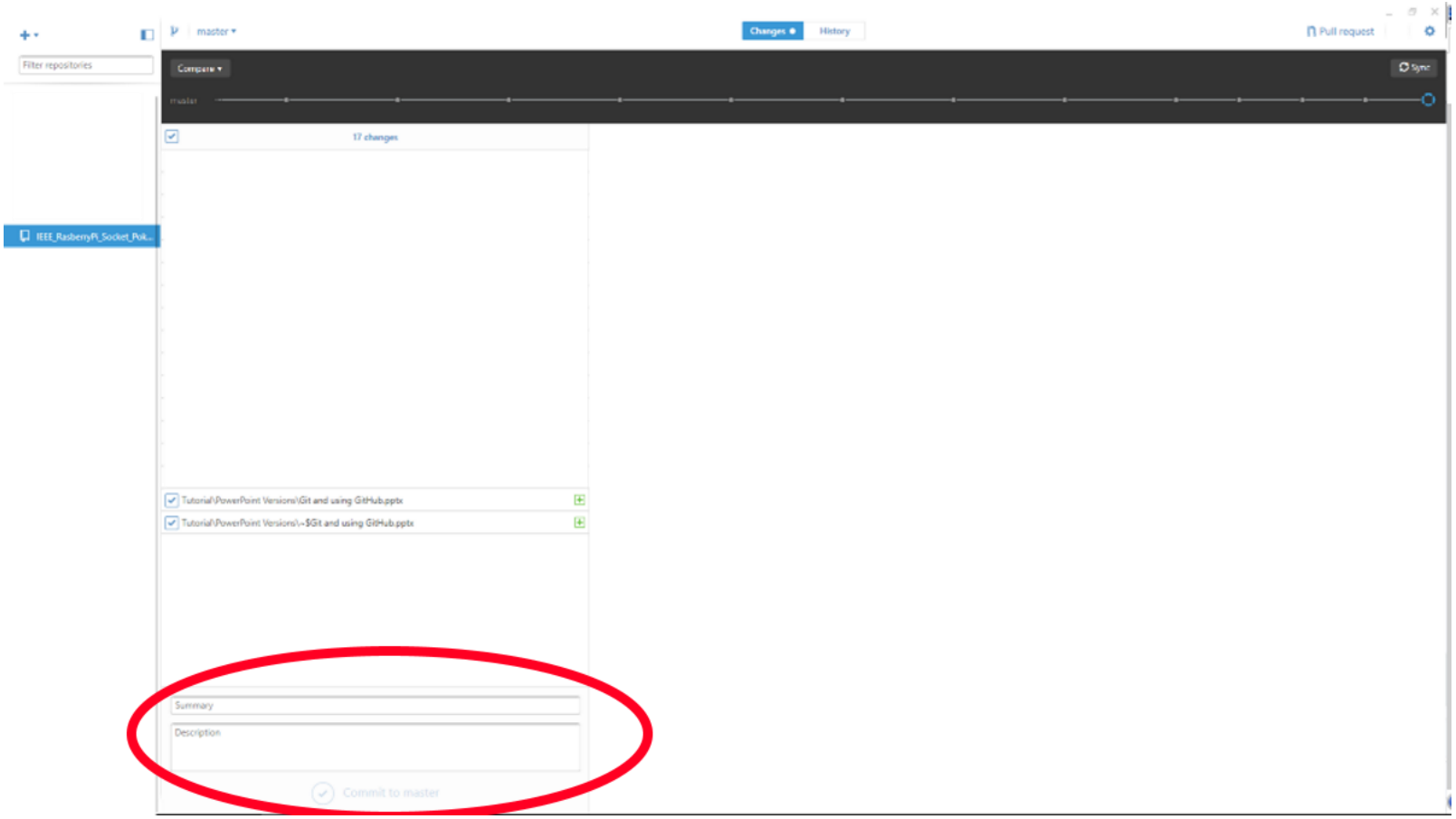
- Click here to sync with Repo
  - Use to get the latest version (ALWAYS Do this before you start!)
  - If you are a Collaborator you can use this to push to Server



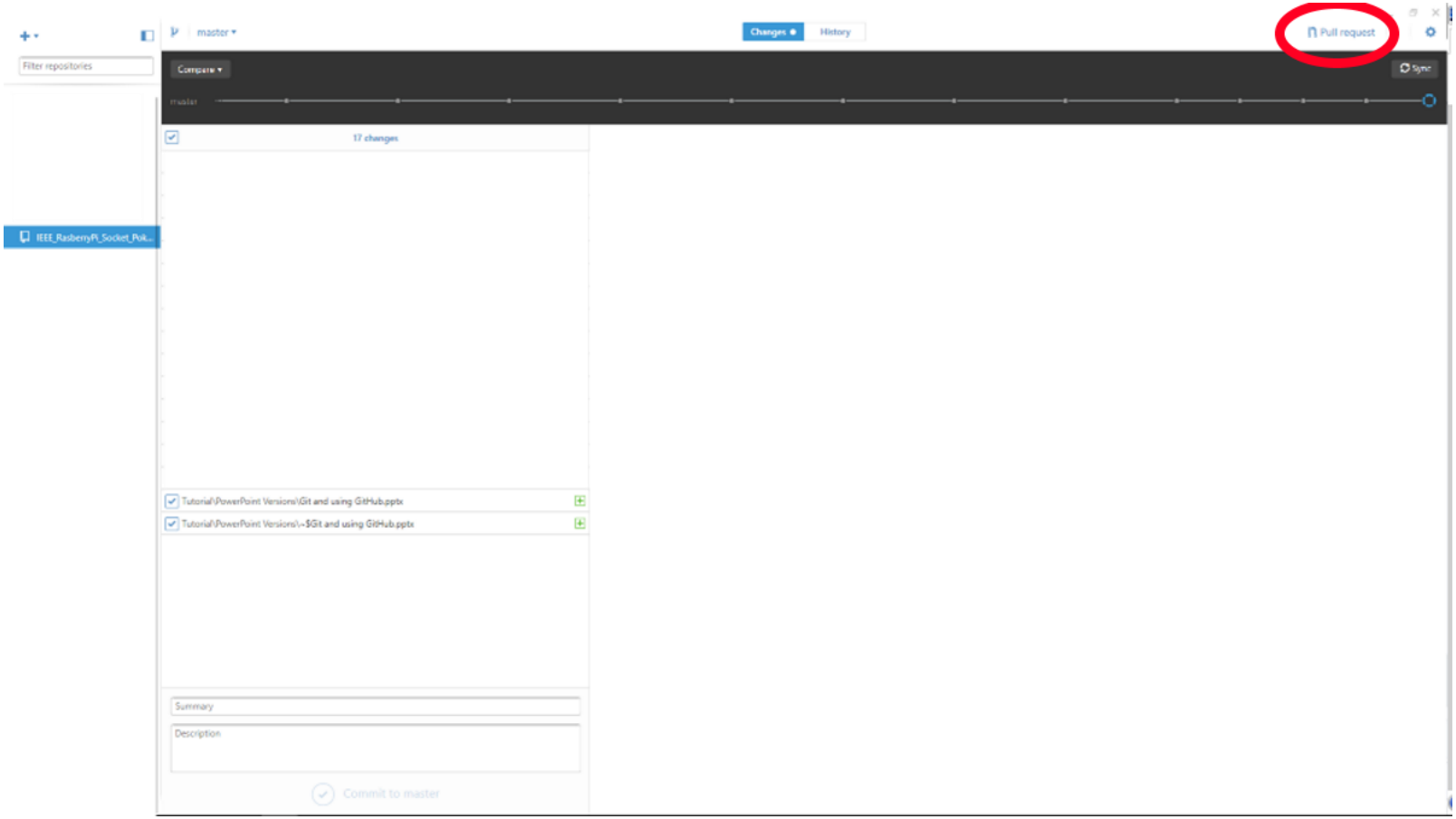
- Either Add a new branch or switch between branches of the project



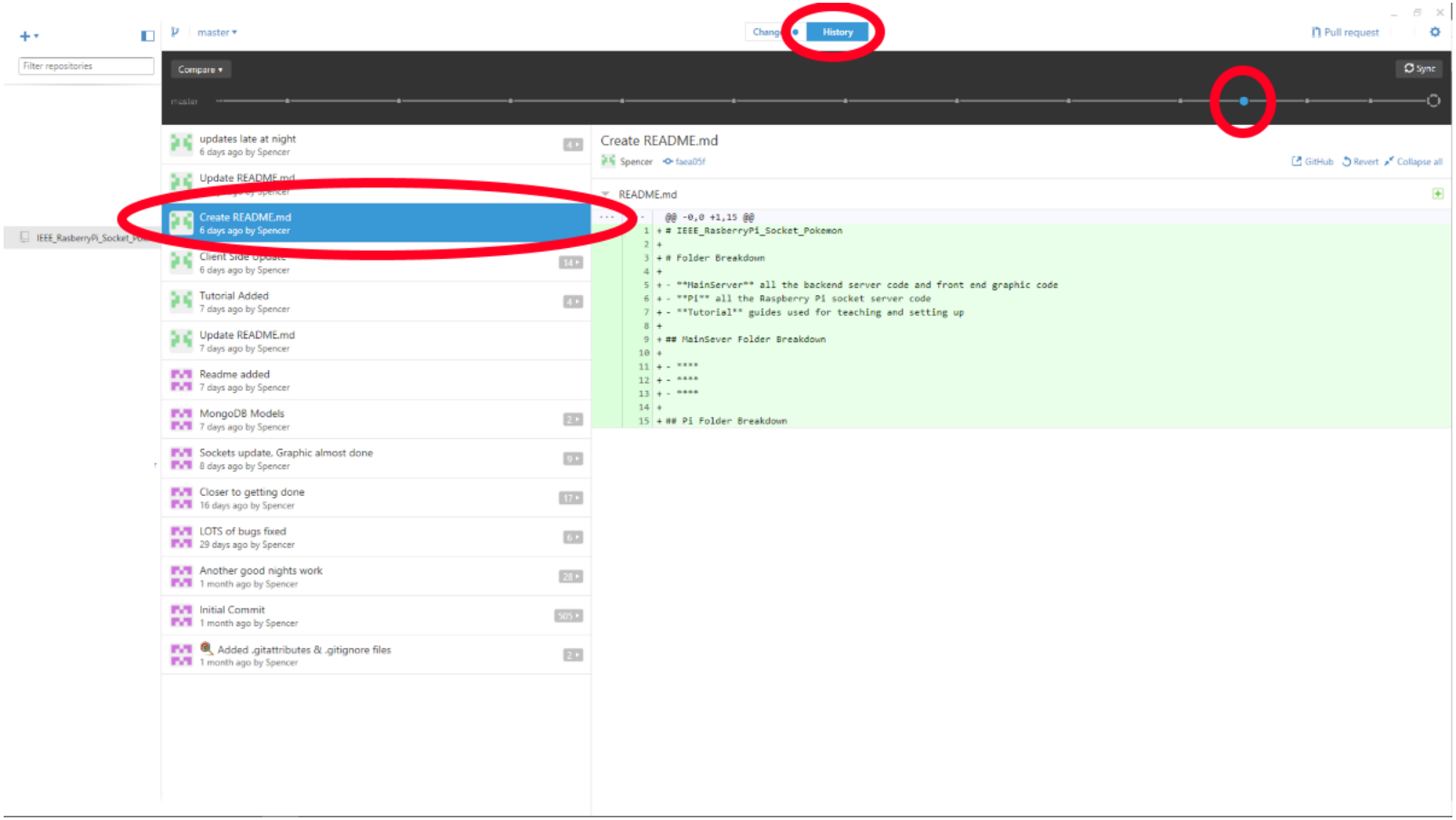
- When in “Change” screen you can select which files you are **Staging** and want to add to the new changes



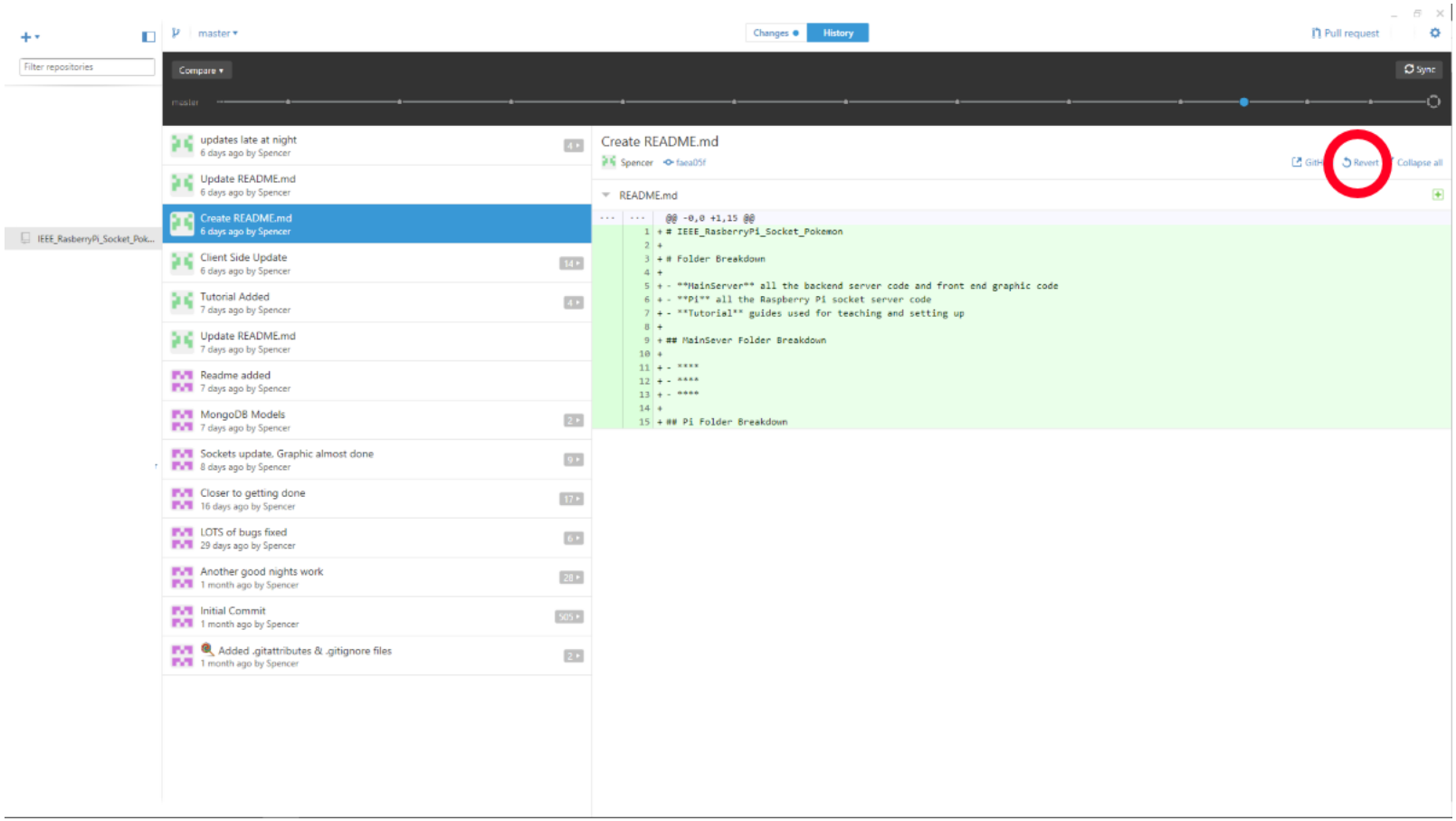
- Add a Summary and Description to your **Commit** and **Commit** the changes
  - Remember, this is only committed on your local machine until you push the changes to the server



- If your code is ready, you can send the **Pull Request** from here



- In the “History” screen you can see all the past **Commits** of the repository



- You can then **Revert** back to that point in the project… maybe start a new branch or see what you did or changed… this is WHY you use Version Control in the first place

Almost Forgot about .gitignore

- Git is not the best with Binary files (.mp3, .pdf, .exe, .FileTypeICannotReadInNotepad)
- When you compile projects or other things that are not made well for git, we use the .gitignore file
- It list all the folders and files that git will not recognize and won't ask you to stage them for committing
- Almost all types of projects have a standard .gitignore template found on GitHub when making a new project

Recap

- Get Repository (**Clone**) or Sync it **(Fetch)**
- Know which **Branch** you are in
- Make your changes
- Set which changes you are **Staging**
- **Commit** the changes
- When ready, Sync or send a Pull Request
- Think how awesome it is you can now work on code projects with someone else, granted they probably didn’t comment it correctly, but that’s a different slide show I am not writing

Best Practice

- These are the best two sites to practice with
  - https://try.github.io/levels/1/challenges/1
  - http://learngitbranching.js.org/

- Here is a repo you can mess with

  https://github.com/sjfricke/IEEE_Practice_Git