Using Git Version Control

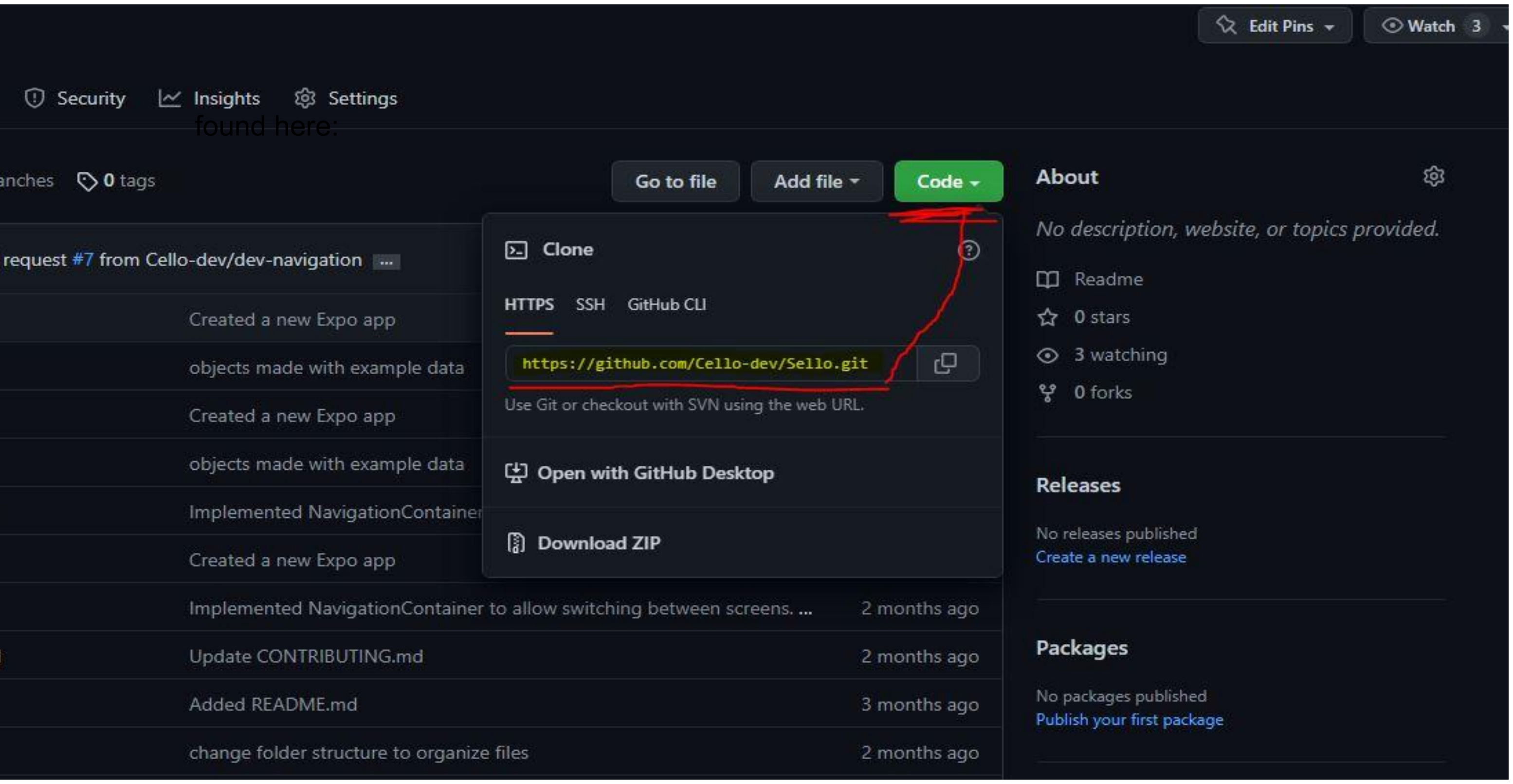
This is to be used for someone who has never used or needs a refresher on how to use Git. This is a comprehensive guide to using Git as a version control system. It will explain the things that are needed to be known from cloning the repository to the process that must be done in order to push it up to the remote repository

1. Cloning the Repository

When Cloning the repository, it is as simple as placing the command

```
git clone REPO_NAME
```

The REPO_NAME is found after clicking the **Green** Code button



found here:

https://github.com/Cello-dev/Sello.git

Use Git or checkout with SVN using the web URL.

request #7 from Cello-dev/dev-navigation ...

Created a new Expo app
objects made with example data
Created a new Expo app
objects made with example data
Implemented NavigationContainer
Created a new Expo app
Implemented NavigationContainer to allow switching between screens. ... 2 months ago
Update CONTRIBUTING.md 2 months ago
Added README.md 3 months ago
change folder structure to organize files 2 months ago

2. Branch before making changes

When working on a new story, make sure that you make a new branch so that it will not impede on the progress already made. This is done by using the Git Bash command

```
git checkout -b BRANCH_NAME
```

If you are changing branches to work on a particular part that is already a work in progress, you would use the same command but omit the *-b* from the previous command

To make sure that the branch that you are looking for is the correct branch, you can use the command

```
git branch
```

This will print a list of all of the branches that you have access to in the repository, there will be one branch that you are working on and that one will be a different color and have an asterix to differentiate from the others

3. Build to make sure, code is working as it is intended

Run the code to make sure that it works, and if there is something wrong, I would contact someone on the team to make sure that they are aware of the issue and have someone figure out the root cause of the problem

4. Code!

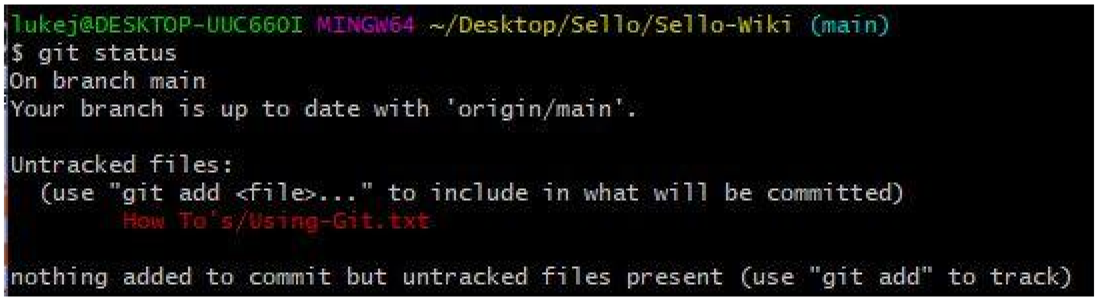
This is the time when you are finally in the right location and it is your time to make your changes that you have made. While doing this remember to make sure that you are commenting and making sure that the code that you are working on is readable to the other people that are working on it. While doing this, also keep in mind that if you are making any functions that you believe have not been documented yet and are likely to be used again, make sure that they get put into the documentation

5. Commit changes

It is good practice to commit your changes after completion of a task.Commit is only happening locally, in order to make sure that it is happening remote you would use the push command, which will be later on. But before you commit you have to stage something for commit
To see the things that are in the staging area as well as the things that have been changed, you use the command

```
git status
```

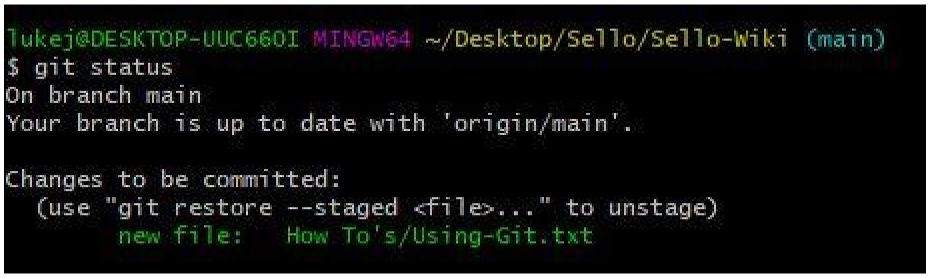
If there are files that are not tracked, then they would look like this:


```
lukej@DESKTOP-UUC660I MINGW64 ~/Desktop/Sello/Sello-Wiki (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        How To's/Using-Git.txt

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

The next thing is to add the files to tracked list by using the command

```
git add <Filename> <filename>....
```

What this will do is put them into the staging area which will be


```
lukej@DESKTOP-UUC660I MINGW64 ~/Desktop/Sello/Sello-Wiki (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   How To's/Using-Git.txt
```

Now that the file is tracked, we can commit it by using the command

```
git commit -m 'Helpful-message'
```

When committing your code even though you may be doing it remotely for now, make sure that the helpful commit message makes sense for what it is that you have done look to the CONTRIBUTING.MD file for more information on what constitutes a good versus a bad git message

6. Merge the remote with the current version

Before pushing the code up to the remote repository, you have to make sure that you get the changes that were pushed into the main branch. This is because there are several people working in the same area of code as you and when they have something that is pushed to main it would make sense the you also have it in your branch, this will make it so that everything is up to date before it is sent up

You would use two different commands the first one:

```
git fetch origin
```

By doing this, it gives you the reference to the remote repository which will be needed for when you merge the code. Which is using the command

```
git merge origin/main
```

In doing this, there is a chance that there may be conflicts these should be delt with then the code that you are working on should then be re-commited

7. Push the code

At this point, all of the code should be ready to be put onto the remote branch. If you are pushing it to a branch and not the main branch(which will most likely be the case) you will be using the command

```
git push origin BRANCH_NAME
```

From this point the code should be found on the remote branch that was specified as *BRANCH_NAME*. You should be ready to request that the code be integrated into the main branch

8. Pull Request

After pushing to the remote branch if you go to that branch you will see that the code is there, right above the code, you will see something that says you should do a pull request, click that

A message should be filled out in the pull request explaining what it is that was done in that branch. You should then let the team know that a pull request is up then they are able to look at the code and see if there are any things that need to be changed if something did need to be changed then, repeat steps 5-7 until the code does not have any issues