Time	~ 1 - 1.5 hours
Learning Goals	• What is GIT and what is it for? • How to use one and work with your friends and peers? • How to generate an SSH key • Understanding and using basic Git commands for code submission • How to write a good Git commit message

In this challenge, we will be learning about Git, a life saving tool in software development. For this challenge, find a partner in your group. We will be doing pair-programming for this round.

Now let's read a story about Tommy and Jane:

Now that you have read Tommy and Jane's story and understand why Git is needed, let's try it out!

Objectives:

This challenge involves using the terminal and your text editor (for example, Sublime or Atom). The given steps will be 'milestones' as you work your way to finish up the challenge. **You'll need to do some research in and out to figure out the WHY and** HOW questions.

1) Fork and clone the Challenge Repository

First, go to this link: https://github.com/NextAcademy/introduce-me-to-everyone.

Fork (go to the top-right hand corner and look for a "fork symbol"). Once you have forked it, pay attention to the URL. You will notice that the repository is now under your name.

Now click on "clone or download". Copy the link given. In your terminal, type the following:

```
git clone https://github.com/your_user_name/introduce-me-to-everyone.git
```

Observe what happens! You should be able find a new folder inside a working folder called "introduce-me-to-everyone" with the repository contents inside it.

So what exactly was happening what you copied the command line in your terminal? We would like you to do some research on this. The keywords to search for are:

1. Repository (remote and local)
2. Local machine
3. `git clone`
4. Fork a repository
5. Clone a forked version of a repository

Discuss your findings with your mentor to see if you are on the right track!

3) Working inside Your Reponsitory

Now go into the working folder. In your terminal, type `cd introduce-me-to-everyone` in terminal to go into the folder (try to start getting used to using the command line from now on!).

Inside the folder, you will find a "my*profile" file. Our task is to fill in all the section with your* answer. Feel free to be creative. For this guide, _to keep changes small, let's fill in only your name. Something like this:

```
================================================================================
What's My Name
================================================================================
I'm the Obi Wan HTML Kenobi, master jedi of frontend
```

Save the file once you're done (you can use Ctrl-S in Sublime to save a file).

Point to discuss:

1. Why small change? Why can't we fill in all the answers and change it one shot?
2. What benefits will you get by doing small changes?
3. How do we define small changes?

Keywords when you talk to your mentor: _managable, trackable, concise, single purpose_

4) Add/Reset

Now we need to add our changes to a "stage", often called "staging". This stage is mainly for small change involving multiple files.

To add a changed file, we'll be using `git add`. To unstage a changed file from staging, we use `git reset`.

Now that you have made a small change to the file by adding your name, type the following the terminal:

```
git add .
```

What do you think happened here? Why do put a "." after `git add`?

Keywords to research on:

1. Staging
2. `git add`

5) Check Status

Okay, we added our changes. Do we have a command to check the current status of Git? Yes, we have, let's bring `git status` on board!

Type `git status` in your console.

Discuss with your partner:

1. What can I learn from the output?
2. How do I see files that are in the staging or what are not?
3. How often I can use `git status`?

6) Commit

Now that we have the correct changes in the stage, it's time to commit. To commit, we'll be using `git commit`. Please take note that there are 4 types of git commit:

1. `git commit`
2. `git commit -a`
3. `git commit -s`
4. `git commit -m`

Your task is to work with your partner and research on each type. In all our challenges, we would like you to use `git commit` and practice writing good commit messages as it will help both you and your team members know what changes you have done to the code.

Here is a good reference on how to write good commit messages.

Once you are done reading, discuss with your partner on what would be a good commit message to write here. Discus it with you mentor too. When you are ready, type `git commit` in your console and then the commit message that you would like to write.

Once you have committed your changes, try typing `git status`. Can it check if a commit is created successfully? What about `git log`? Research again =)

7) Repeat Step #3 again for Other Fields

Hooray! You have committed a small change inside your git. Let's repeat steps #3 to #6 again for the other fields in the file.

8) Push it back to Remote Repository

Once you're happy with the profile file, it's time to push to remote repository to show your changes. You'll be using `git push` to perform such push to remote repository.

Once done, check with `git status` again to see if the remote is working. You should work with your partner on how to check git status.

Wrap Up

If your step 8 is working fine and your remote repository is working, congratulations! You've just experienced the very basics of the git. It's good enough to move forward until the next stage. We will be looking at how to do git branching next!

Commonly used git commands:

1. `git clone`
2. `git add`
3. `git commit`
4. `git push`
5. `git reset`
6. `git status`

Useful References

1. Git
2. try.github.io
3. What is an SSH key?
4. GitCommitMessages

5. How to Write a Git Commit
6. The Art of the Commit