Objectives

* Initialize a local Git repository.
* Learn to perform basic Git operations: add, status, commit, and more.

Instructions

Initialize a local repository

* In your bash terminal, navigate to the empty **git-test/** folder inside your **NucampFolder/**(which should be on your Desktop). It's important that you make sure to be in the **git-test/** folder before you proceed.
* If you're not sure how to get to a specific folder inside your terminal, revisit the lesson on [Using the Bash CLI](https://learn.nucamp.co/mod/book/view.php?id=7282&chapterid=7263).
* When you're in the **git-test/** folder, your terminal prompt should look similar to this on Windows (you may or may not have OneDrive in your path, depending on your setup):

Or like this on macOS:

* It is **VERY IMPORTANT** that you are in your **git-test/** folder. Make sure it says **git-test**in your command line's prompt as above (though yours will look different depending on your computer name, user name, and OS). If you are not 100% sure that you are inside the **git-test/** folder, please take a screenshot of your terminal and ask in the **#javascript** chat channel if it looks correct.
* Initialize an empty Git repository with this command:

git init

Check your Git repository status

* Type the following into your bash terminal to check your Git repository's status:

git status

Create a new file using Visual Studio Code

* Open VS Code in this folder by typing:

code .

* + Note the **space**and **dot**following the word **code** above - do not omit them!
  + If you are on macOS and the above command did not work for you, first make sure that your **Visual Studio Code.app** file is in your **Applications**folder and not your Downloads folder or anywhere else, then follow the instructions under "Additional Step for macOS" [here](https://learn.nucamp.co/mod/book/view.php?id=7281&chapterid=7250). If it still does not work, close and reopen your terminal (to the**git-test/**folder) and try it again.
* In VS Code, create a new a file named **index.html** in this folder, and add the following basic HTML code to this file:

<!DOCTYPE html>

<html>

<body>

<h1>This is a Header</h1>

</body>

</html>

* Again, check your repository status and take note of what it says:

git status

Add files to the staging area

* To add files to the staging area of your Git repository, type:

git add .

* Again, check your repository status and take note of what it says:

git status

First commit to the Git repository

* To commit the current staging area to your Git repository, type:

git commit -m "first commit"

* Again, check your repository status and take note of what it says:

git status

Further demonstration of Git

* For the purposes of this course, adding and committing your code to your local repository after making significant changes is the most you will be asked to do, and even that will be optional.
* We will not require you to do anything else with Git in the rest of this course, because Git can be a complicated topic and we want you to be able to focus on learning the main course content.
* However, please follow along the below instructions for a brief demonstration of how Git can access previous commits.

Second commit

* Now, modify the**index.html**file as follows, inserting a paragraph element:

<!DOCTYPE html>

<html>

<body>

<h1>This is a Header</h1>

<p>This is a paragraph</p>

</body>

</html>

* Make sure this file is saved, then in your bash terminal, type:

git add .

git commit -m "Second commit"

Third commit

* Once more, modify the**index.html**file as follows:

<!DOCTYPE html>

<html>

<body>

<h1>This is a Header</h1>

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

</body>

</html>

* Make sure this file is saved, then in your bash terminal, type:

git add .

git commit -m "Third commit"

Check the log of Git commits

* To check the log of the commits to your Git repository, type:

git log --oneline

Checking out a file from an earlier commit

* To check out**index.html** from the second commit, find the number of the second commit using**git log**, and then type the following at the prompt:

git checkout <second commit's number> index.html

* Notice in VS Code how the code has changed.
* Try it again with the first commit -- use **git checkout** <first commit's number>**index.html**. (Without the dot at the end of **index.html**.)

Resetting the local Git repository

* To discard the effect of the previous operation and restore index.html to its state at the end of the third commit, type:

git reset HEAD index.html

* Then type the following at the prompt:

git checkout -- index.html

* You can also use**git reset** to reset the staging area to the last commit without disturbing the working directory.

Summary

* In this lesson, you have learned how to initialize a local repository (local meaning, on your own computer), along with how to use basic commands to add files to staging then commit them, view the git log, check out earlier commits, and reset the repository.