**SPRINT 1: Command line - Primer**

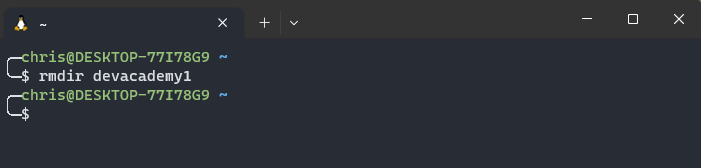
Activities

# STRETCH

## **Remove or copy a directory**

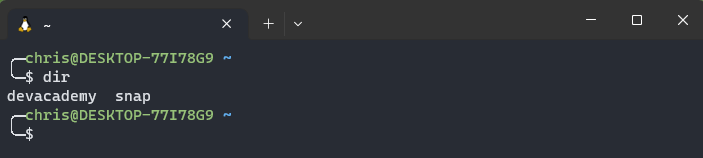
**Remove a directory (devacademy1)**

**Command**: rmdir devacademy1  
**Output**: output displays that when executing the terminal command rmdir devacademy1 the output process displays as indicated in the image below.



**Check to see if command executed successfully**

**Command**: dir   
**Output**: output of the dir command allows us to view directories within the current directory that you are working in. the rmdir devacademy1 command successfully removed the directory devacademy1



**Copy a directory**

To be able to copy a directory we need to know what directories are in the current directory we are working in by using the dir command

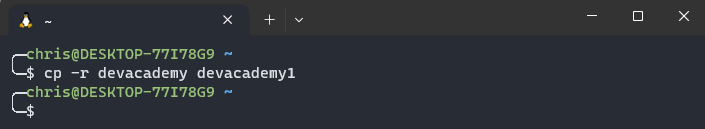
**Command**: dir

**Output**: command executed displays devacademy and snap directories. We are going to copy the devacademy directory



**Command**: cp -r devacademy devacademy1

**Output**: devacademy1 directory copied successfully



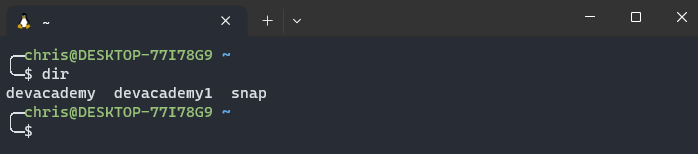
**What does the -r command mean?**

The -r command stands for recursive this command can be used with other commands e.g., cp, rm, mv etc. this command can be also applied to directories with subdirectories and files within the directory. But its good to be cautious when using this command as it permanently removes subdirectories and files within a directory if using the rm command.

**Check to see if devacademy1 directory has been copied**

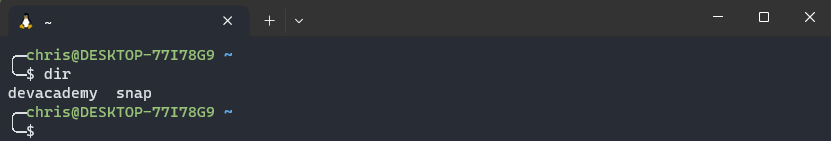
**Command:** dir

**Output**: devacademy1 directory has been copied successfully



**Remove directory (devacademy1)**

**Command:** rm -r devacademy1

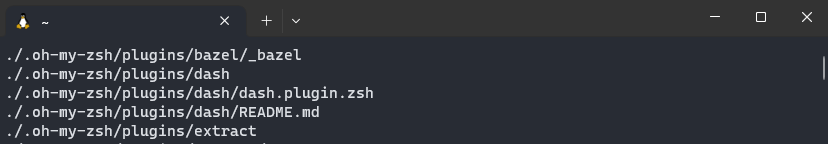
**Output**: devacademy1 directory has been successfully removed

## **Find and look inside files**

**Find all files within the current directory**

**Command:** find .

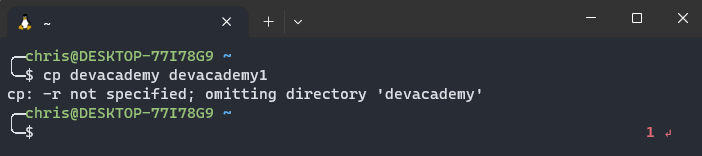
**Output:** the find . command finds all files within the current directory



Observations

**Copying directory devacademy**

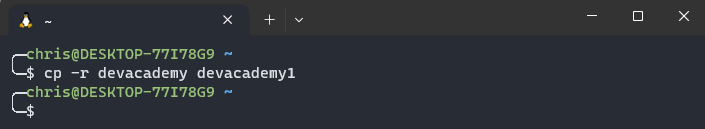
**Command**: cp devacademy devacademy1  
**Output**: output displays that -r was not specified; omitting directory ‘devacademy’. My assumptions are is that we needed to add the -r along with the cp command in order for the copy to work successfully.



**Attempt 2**

**Command**: cp -r devacademy devacademy1

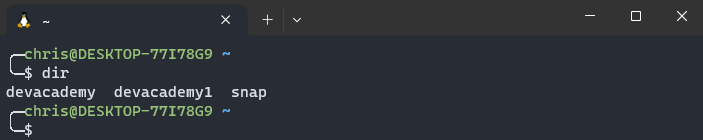
**Output**: command executed successfully



**Check to see if** cp -r devacademy devacademy1 **command successfully executed**

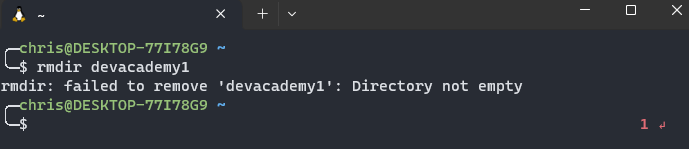
**Command**: dir

**Output**: devacademy1 directory created



**Removing directory (devacademy1)**

**Command**: rmdir devacademy1  
**Output**: output displays failed to remove because the directory is not empty. I now understand that doing the command rmdir devacademy 1 wont work if the directory is not empty so we are going to attempt the -r command with rm command to see if it works.

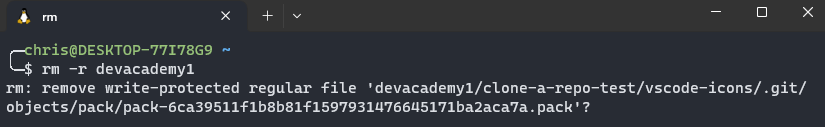


**Remove directory devacademy1 using rm -r command**

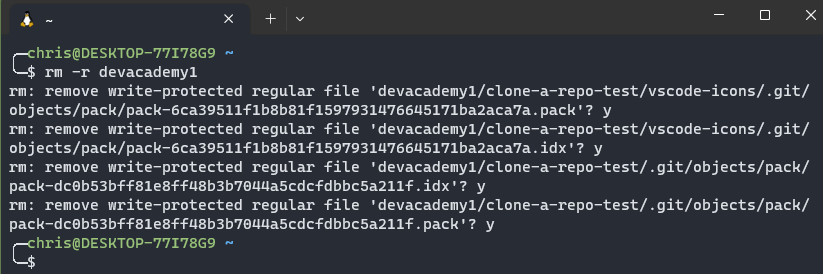
**Attempt 2**

**Command**: rm -r

**Output**: out displays a caution message asking to remove directory as indicated below. At the end of the question mark type the y key. Y stands for yes to confirm the removal of the subdirectories within the devacademy directory



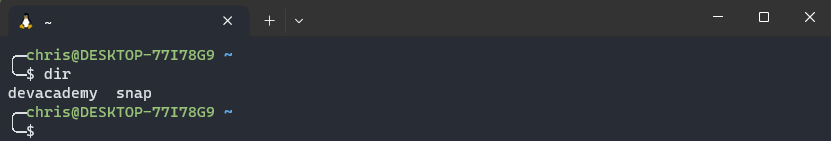
After acknowledging the caution messages to remove subdirectories and files within the devacademy1 directory command prompt goes back to the home position assuming it has removed the devacademy1 directory.



**Check to see if devacademy1 directory has been removed**

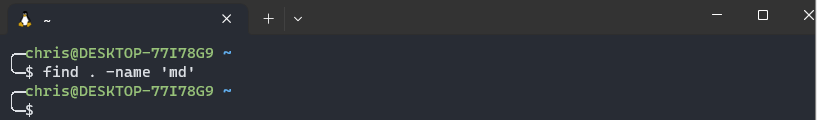
**Command**: dir

**Output**: devacademy1 directory successfully removed



Find

**Command**: find . -name ‘md’  
**Output**: command unsuccessful as I did not find any files with the letters md inside it



**VERSION CONTROL WITH GIT**

Activities

## What is version control?

Allows tracking on what you worked with and what your peers have worked on.

Version control also gives you the use so that if you and your peers are working on the same file you will not clash and overwrite each other’s changes within the same file.

Although you and your peers are working on the same file version control will send out a caution message stating that you are going to overwrite your peers’ changes within the same file.

Version control allows you to work together with your peers in a more efficient manner.

## What is Git?

Git is commonly used for software development and has become one of many most popular version control systems worldwide.

**Key concepts using git**

* **Repository** – a collection of files and revision history is obviously called repository also known as repo, it includes project files and metadata, as well as history of all file modifications.
* **Commit** –commit shows the repository (repo) as it was at the time of commit. It provides a commit hash, a special identifier for tracking changes made to files
* **Branch** – A distinct line for development in a repository is called a branch. You can work on features or bug fixes without impact on the primary codebase. When work is finished, the commits from each branch can be merged back into the main branch (often referred to as "master" or "main").
* **Clone** - A repository's full contents, including all of its files, branches, and commit history, are copied locally when a repository is cloned. Thus, you can work on the project locally and afterwards sync your changes with the remote repository.
* **Pull** - Pulling is the process of obtaining the most recent updates from a distant repository and integrating them into the active branch. It adds the most recent commits made by other users to your local repository.
* **Push** - Sending your local commits to a remote repository is known as pushing. You can make your changes available to the team and distribute them to others using this tool.
* **Merge** - Changes made in various branches are combined into one branch through merging. It transfers the modifications made to one branch to another.

Along with numerous graphical user interfaces (GUIs) and integrated development environment (IDE) integrations, Git offers a robust command-line interface (CLI). Due to its adaptability, speed, and comprehensive feature set, it has emerged as a crucial tool in contemporary software development.

## What is GitHub?

A web-based hosting service for Git repositories is called GitHub. It gives programmers a platform for project collaboration, code sharing, and Git version control management.

**Key concepts using GitHub**

* **Repository Hosting -** Users can build and host Git repositories on GitHub's platform. Both private repositories, which need permissions to access, and public repositories, which are open to everyone, can be created.
* **Collaboration and Social Features -** Developer collaboration is made possible via GitHub's technologies. You may control user access rights, invite more people to participate in your projects, track and communicate about coding-related difficulties, and more. Pull requests, a feature of GitHub that makes it easier for users to discuss and evaluate code before integrating changes into a repository, are another functionality.
* **Issue Tracking -** Users can create, manage, and discuss issues linked to a repository using GitHub's issue tracking system. Issues can be used to discuss any subject related to the project, report errors, or suggest new features.
* **Forking and Pull Requests -** A feature called forking enables users to make a unique copy of a repository. They are able to freely experiment with changes thanks to this without having an impact on the original repository. Changes made in a forked repository can be suggested back to the original repository via pull requests. Code review and collaboration are made easier by this procedure.
* **Version Control and History -** GitHub makes advantage of Git's powerful version control system to let users easily explore a project's history and track file changes. Viewing commits, contrasting several file versions, and maybe rolling back changes are all possible.
* **Continuous Integration and Deployment -** GitHub offers CI/CD (continuous integration and deployment) interfaces with a number of services. This makes it simpler to maintain and release high-quality code by enabling developers to automate testing, build procedures, and deployment pipelines.

Both free and premium plans are accessible on GitHub, with the higher ones offering more functionality and storage space. It has grown to be a well-liked platform for open-source initiatives, enabling developers everywhere to cooperate, contribute, and present their work.

Observations

## What is version control?

Information about version control was gathered in the following link <https://www.youtube.com/watch?v=uUuTYDg9XoI> you could also find information such as

* Source codes
* Version control
* What is distributed

This information is important when dealing with tasks relating to version control with Git.

## What is Git?

Information about git was gathered in the following link <https://www.simplilearn.com/tutorials/git-tutorial/what-is-git#what_is_git> this link also shows everything you need to know and commands and how they work in relations to git

## What is GitHub?

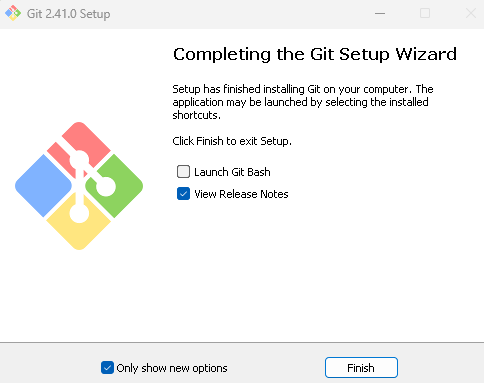
Observations for task was gathered in the following link <https://www.techtarget.com/searchitoperations/definition/GitHub> this link shows you everything you need to know about GitHub and getting started using the software.

**VERSION CONTROL WITH GIT**

Activities

## Version Control with Git – Course

## Install git

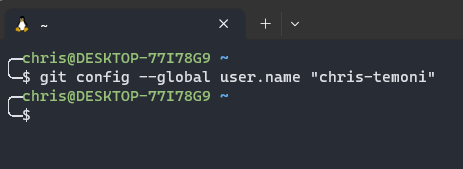


Installation of git completed click on finish. See observations for installation process

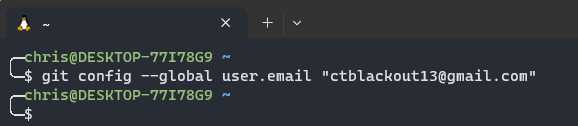
## First time git configuration

**Command:** git config –global user.name “chris-temoni”

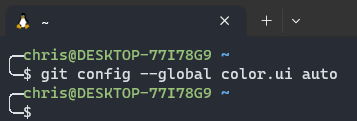
**Output**: output sets up git with my name



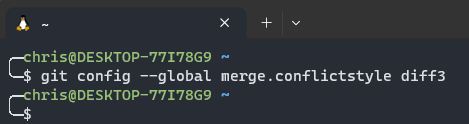
**Command**: git config –global user.email [ctblackout13@gmail.com](mailto:ctblackout13@gmail.com)  
**Output**: output sets up git with email



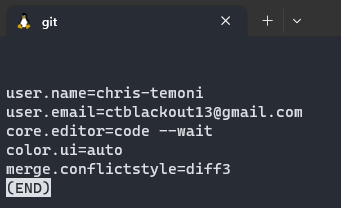
**Command**: git config –global color.ui auto  
**Output**: this command makes sure that git output is colored



**Command**: git config -global merge.conflictstyle diff3

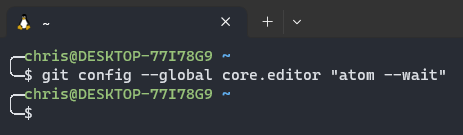
**Output:** Output displays the original state in a conflict

**Command**: git config --list

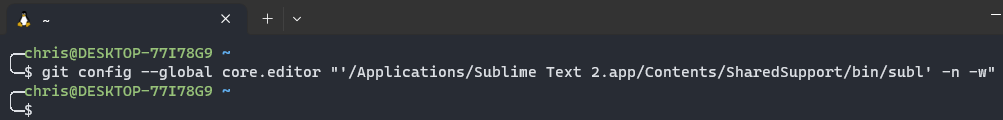
**Output**: displays the config setup that has just been made

**Command**: git config --global core.editor "atom --wait"

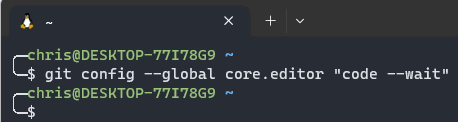
**Output**: this command sets up the atom editor



**Command**: git config --global core.editor "'/Applications/Sublime Text 2.app/Contents/SharedSupport/bin/subl' -n -w"

**Output**: command sets up the sublime text

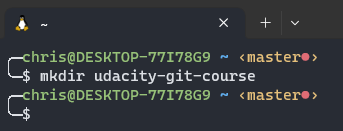
**Command**: git config --global core.editor "code --wait"

**Output**: command executes Vscode setup

## Create a Repo from scratch

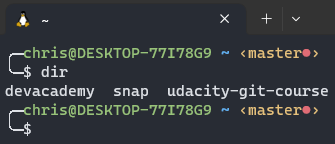
**Create a directory**

**Command**: mkdir udacity-git-course  
**Output**: execution of command indicates that the udacity-git-course has been created



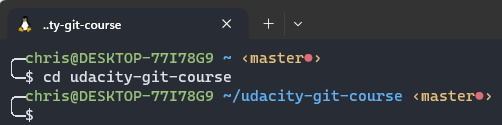
**Check to see if udacity-git-course directory has been created**

**Command**: dir   
**Output**: udacity-git-course directory successfully created



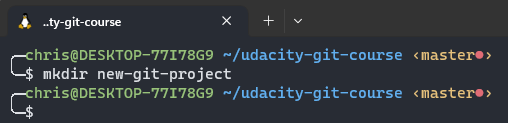
**Create a directory within the udacity-git-course directory**

**Command:** cd udacity-git-course  
**Output**: we are now in the udacity-git-course directory. We are going to use this directory to create another directory inside it.

****

**Create new-git-project directory**

**Command**: mkdir new-git-project  
**Output**: directory new-git project has been created inside the udacity-git-course directory

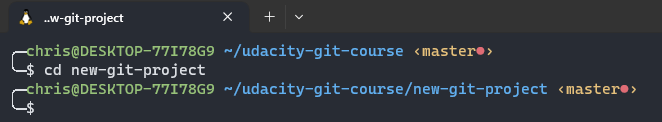


**Check to see if directory has been created**

**Command**: dir  
**Output**: directory new-git-project has been created successfully

**Move into the new-git-project directory**

**Command**: cd new-git-project  
**Ouput**: upon executing the command we have successfully moved to the new-git-project directory



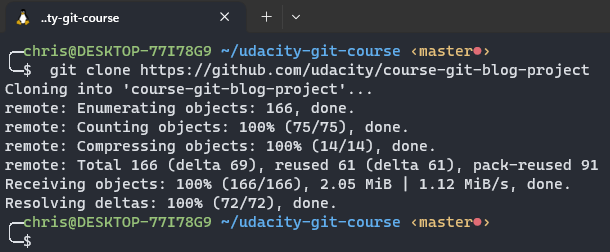
**Git init command**

**Command**: git init  
**Ouput**: the git init command initializes an empty Git repository in the new-git-project directory



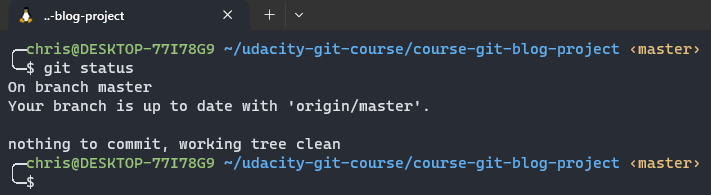
**Cloning blog repository**

**Command**: git clone https://github.com/udacity/course-git-blog-project  
**Output**: command is used to copy blog project repository into course-git-blog-project folder inside the udacity-git-course directory

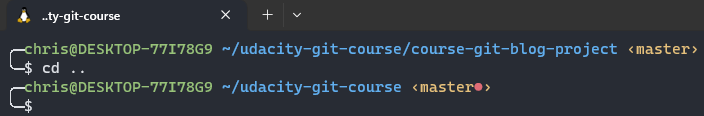


**Git status**

**Command**: git status  
**Output**: command displays that branch is up to date with origin/master nothing to commit, working tree clean (see in depth information in observations)



**Use git status in the new-git-project directory**

**Command:** cd .. **Ouput:** output takes you back to previous directory

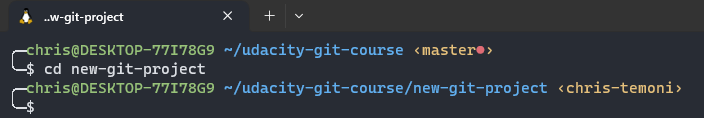
**Check to see if new-git-project is in the udacity-git-course directory**

**Command:** dir **Output:** the new-project directory is in the udacity-git-course directory

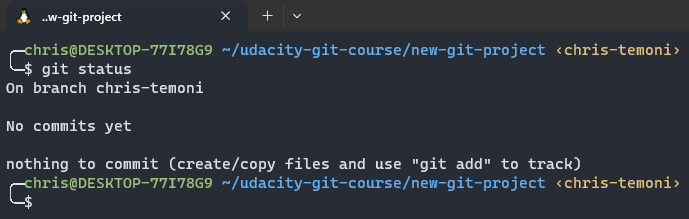


**Change to the new-git-project directory**

**Command**: cd new-git-project directory  
**Output**: output displays that we are now in the new-git-project directory

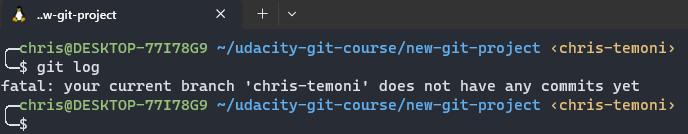


**Execute the git status command**

**Command**: git status  
**Output**: output displays on branch chris-temoni No commits yet nothing to commit (create/copy files and use “git add” to track). More information can be found in observations.

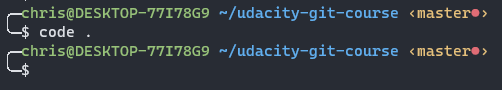
**Git log**

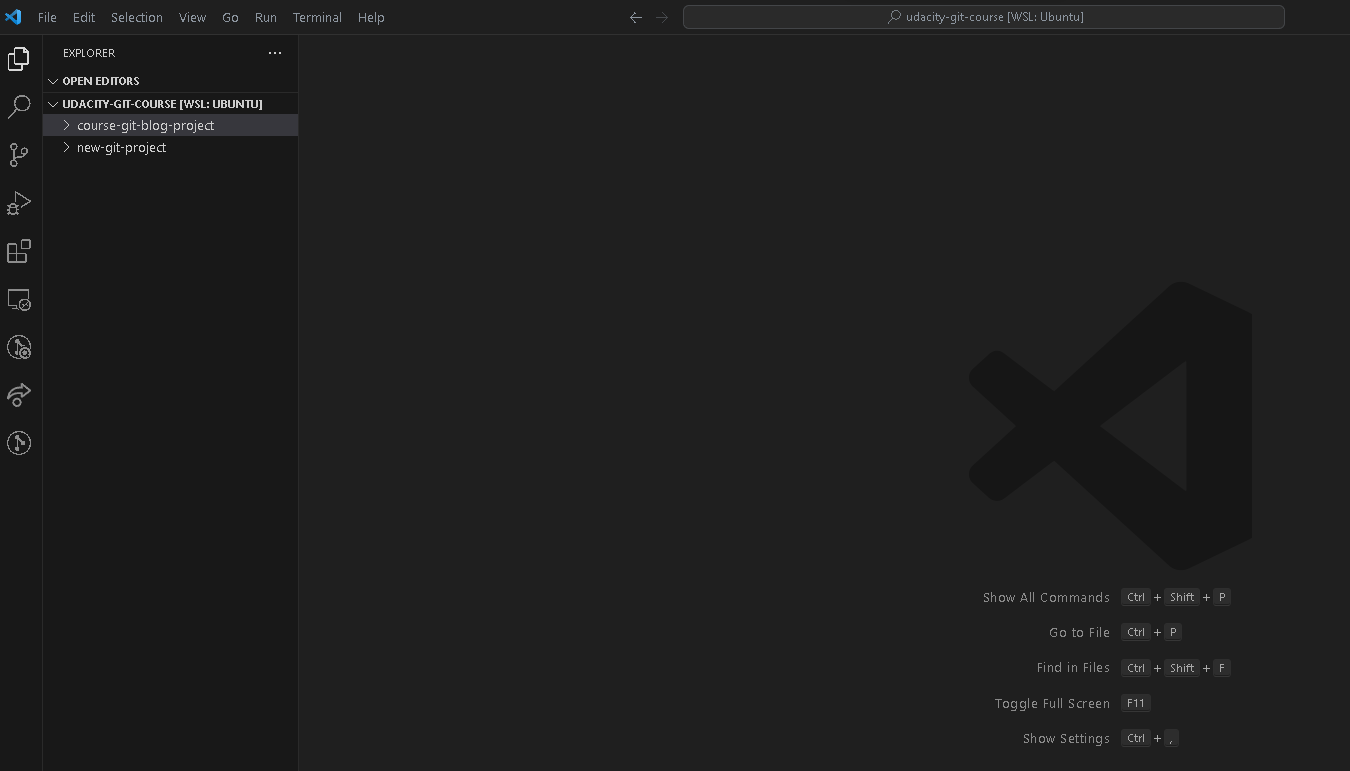
**Command:** git log  
**Output**: output displays current branch does not have any commits.



**Access Vscode through terminal**

**Command**: code .  
**Output**: output of the command opens up Vscode.



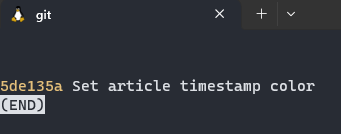
**Output**:

**Execute the git log command in the course-git-blog-project directory**

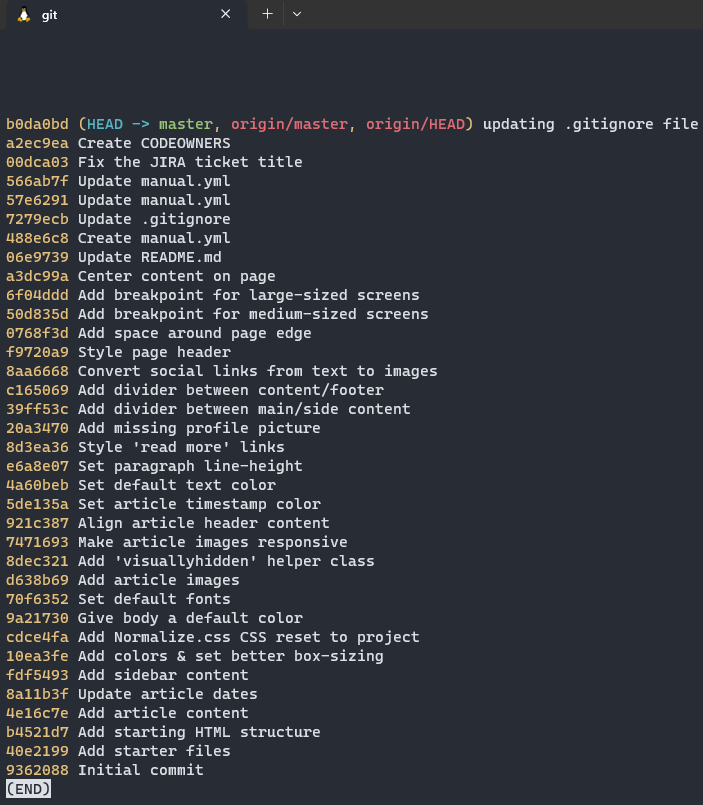
**Command**: git log  
**Output**: git log displays the git sha, the author of the commit the date, time and methods for the command.

**Use git log to find the commit that has the message Set article timestamp color. Which commit belongs to that SHA?**

**Command**: git log --oneline --grep="Set article timestamp color"  
**Output**: output displays sha



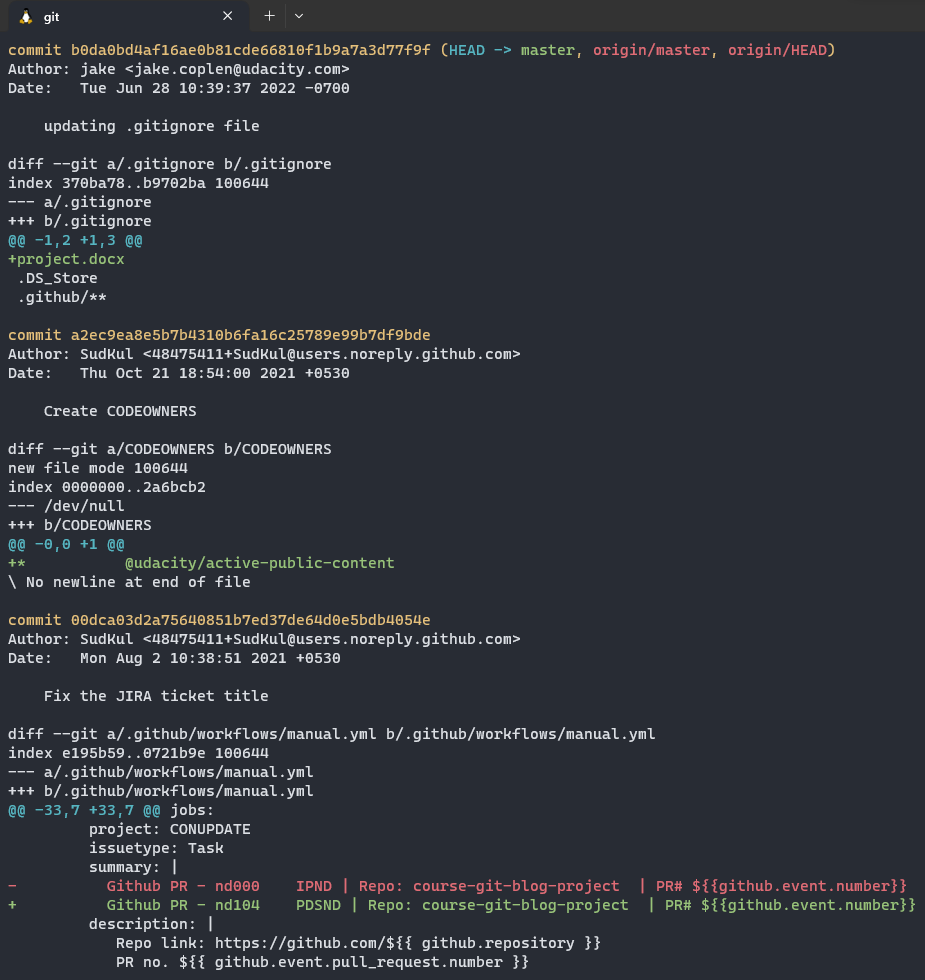
**Git log --one line**

**Command**: git log --one line  
**Output**: output displays that it lists one commit per line first 7 characters of the sha for the commit and shows the commits messages. This is a good command for if you want to quickly review a repository commit

**Git log --stat command**

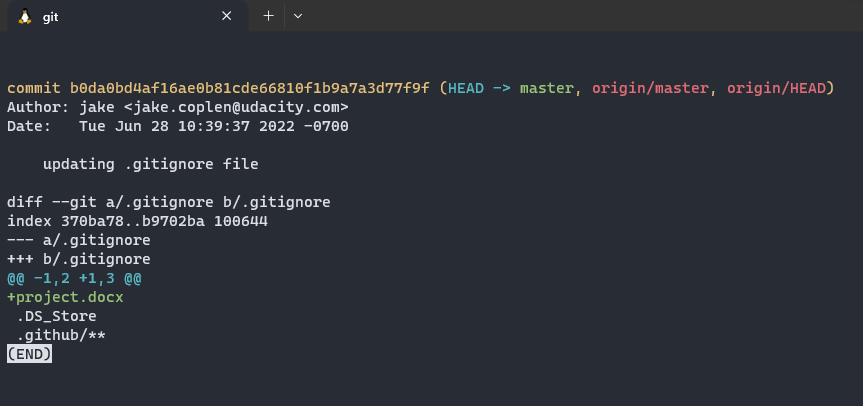
**  
Command:** git log –stat  
**Output**: this command displays the file(s) that have been modified, displays the number of lines that have been added/removed, and displays a summary line with the total number of modified files and lines that have been added/removed

**Viewing changes**

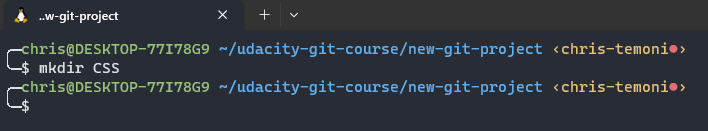
**Command**: git log -p  
**Output**: command displays the files that have been modified, displays the location of the lines that have been added/removed, and displays the actual changes that have been made

**Git show command**

**Command**: git show  
**Output**: git show command only displays one commit, the author, the date, the commit message and the patch information.

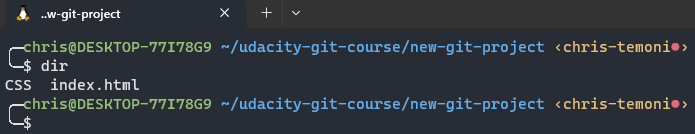


**Create a CSS folder and create app.css inside of the CSS folder**

**Command**: mkdir CSS  
**Output**: CSS folder has been created

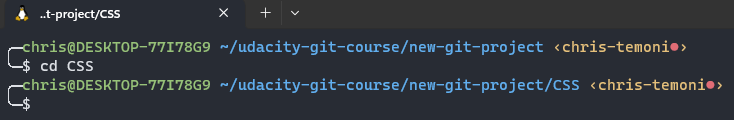
**Check to see if folder has been created**

**Command**: dir   
**Output**: CSS folder created successfully



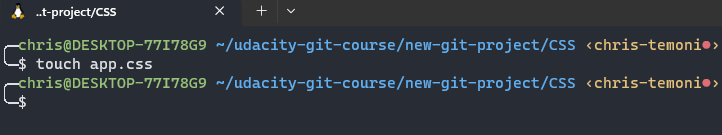
**Create apps.css file inside the CSS folder**

**Command**: cd CSS  
**Output**: now we are in the CSS folder. We are going to create app.css file within the CSS folder



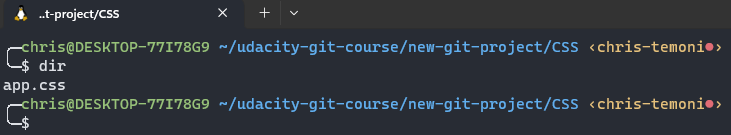
**Create app.css folder**

**Command: touch apps.css  
Output:** app.css file created



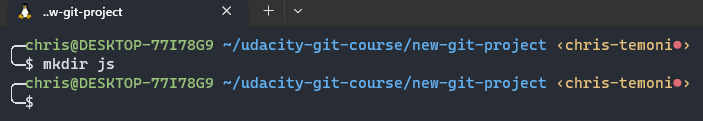
**Check to see if app.css has been created**

**Command: dir  
Output:** app.css file successfully created

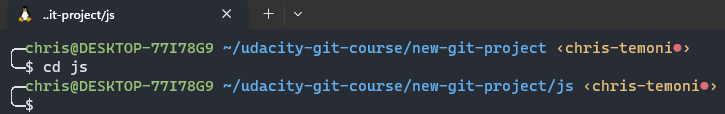


**Create js folder and create app.js inside of the js folder**

**Command**: mkdir js  
**Output**: js folder created

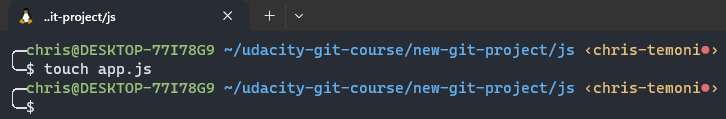


**Navigate to the js folder and create app.js file**

**Command**: cd js  
**Output**: we are now in the js folder

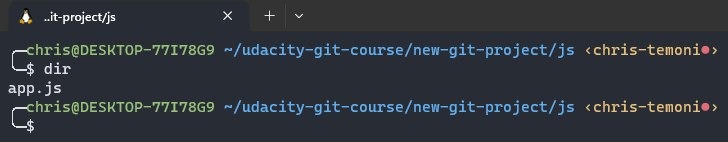
**Create app.js file inside js folder**

**Command**: touch app.js  
**Ouput**: app.js file has been created



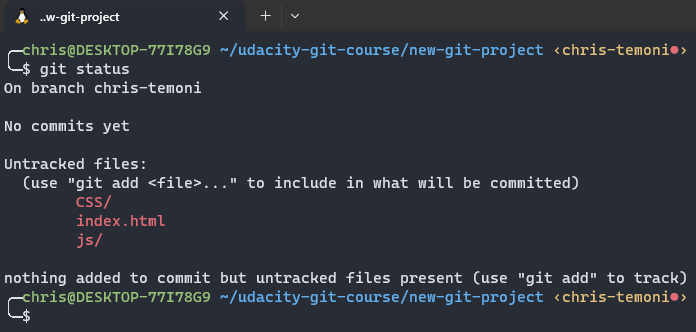
**Check to see if file has been created**

**Command**: dir  
**Output**: app.js file successfully created



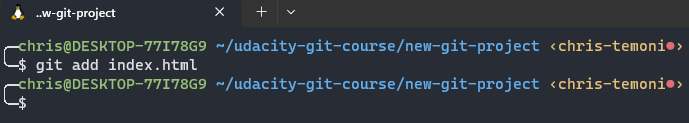
**Git status command**

**Command**: git status  
**Output**:



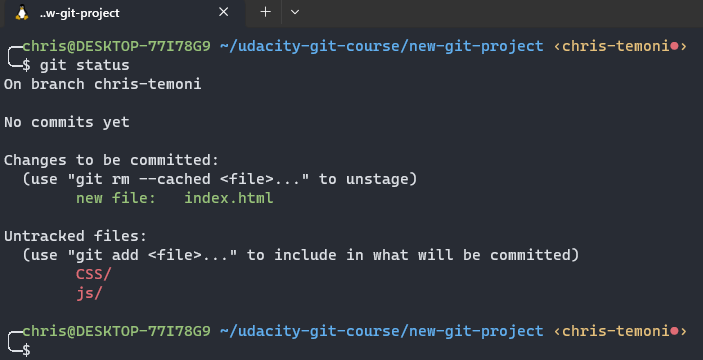
**Staging files**

We are going to move the created folders and files to the staging index  
  
**Command**: git add index.html  
**Ouput**: index.html file was moved to the staging index



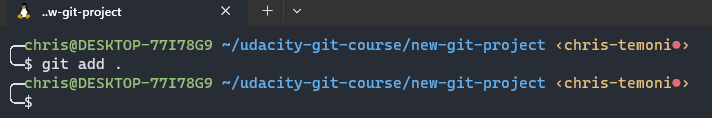
**Check to see if index.html file has been moved to staging index**

**Command**: git status  
**Output**: index.html file has been moved successfully index.html is now in the changes to be committed category



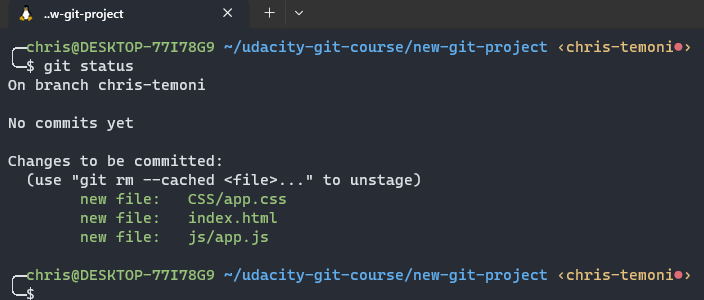
**Stage remaining files using the period . command**

**Command:** git add .  
**Output**: the git add . command added the remaining files NOTE: use the git add . command if you want to add all files having files you do not want to add will add them if they are in the directory so use the command carefully



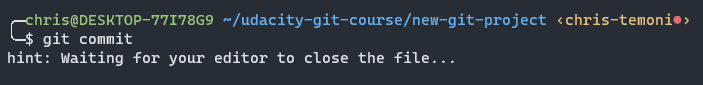
**Check to see if remaining files have been moved**

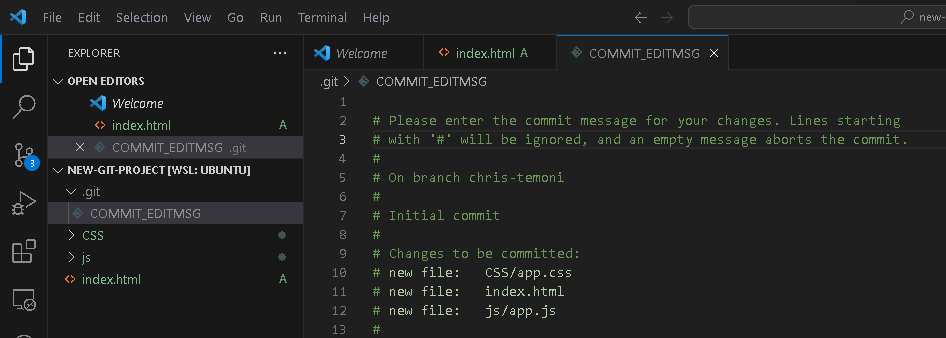
**Command**: git status  
**Output**: remainder of the files have been successfully moved



**Make a commit**

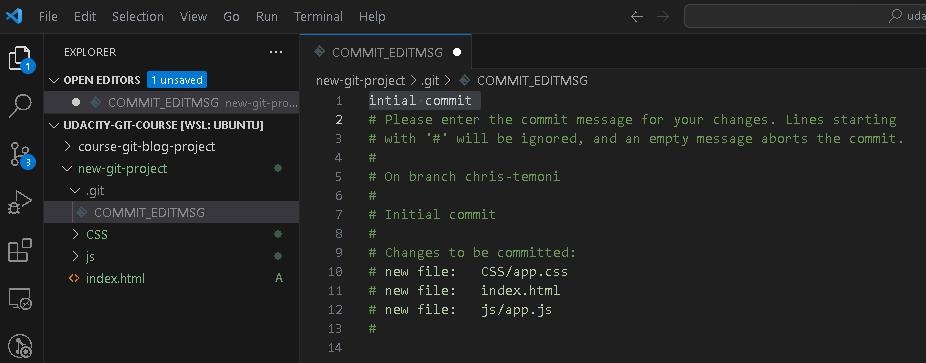
**Command**: git commit  
**Output**: output displays that the command opens up Vscode letting us know that the commit has successfully worked





Finalizing the commit

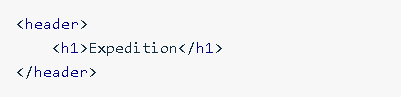
Inside the editor in line 1 add comment initial commit save the changes then close the editor



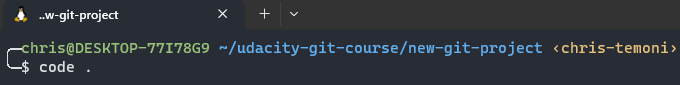
After the above process navigated back to the terminal and information showed as indicated in the image below this is stating information about the commit



**Additional commands**

**Command**: git commit -m “initial commit”  
**Output**: this command will automatically write a message for you (as mentioned above) will bypass the above process.   
  
**Second commit – add changes**add the following into the index.html file

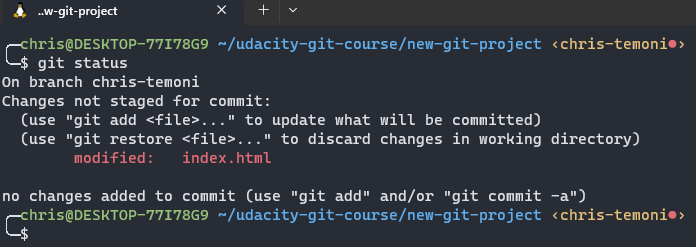
**Command**: code .  
**Ouput**: command will open up the editor





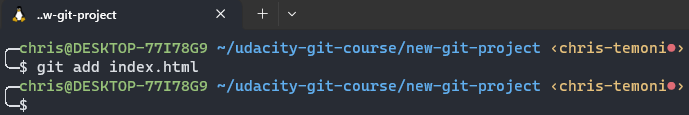
Add the above information in the body section inside the index.html file. After you have input the following code save the changes and close the editor.

**Check to see if changes have been made to the index.html file**

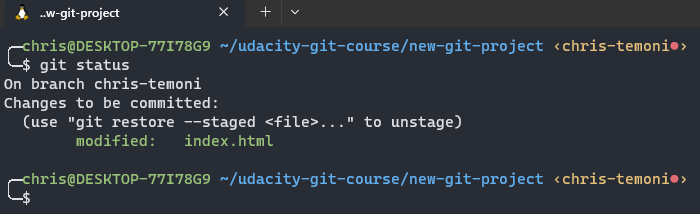
**Command:** Git status **Output:** command displays that changes not staged for commit. section that includes the modified index.html file

**Git add**

**Command**: git add index.html  
**Output**: the modified index.html file has been moved



**Check to see if it has been moved   
Command:** git status  
**Output**: modified index.html file has been moved



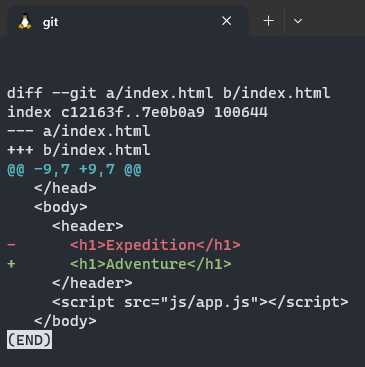
**Second commit**

**Command**: git commit -m “add header to blog”  
**Output**: modified file has been committed successfully



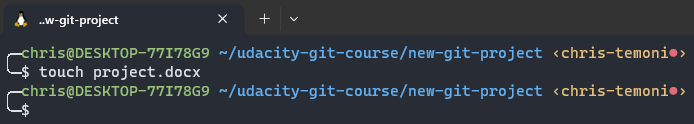
**Git diff**

**Command:** git diff  
**Output**: command displays files that have been modified, the location of lines that’s been added or removed, and changes that have been made

****

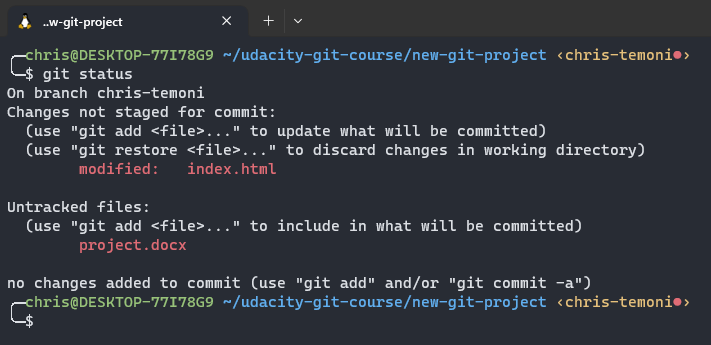
**Create project.docx file in the new-project directory**

**Command:** touch project.docx **Ouput:** project.docx file has been created

****

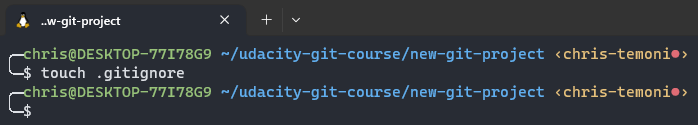
**Run the git status command to show list of files**

**Command: git status  
Output:** output displays new word document in gits untracked files section



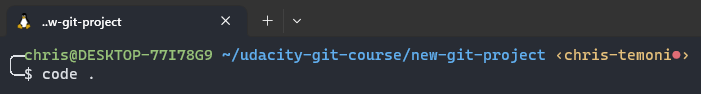
**Create a .gitignore file**

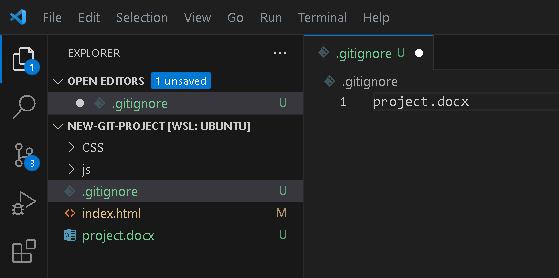
**Command:** touch .gitignore **Output:** the .gitignore file has been created

****

**Add the line project.docx in the .gitignore file**

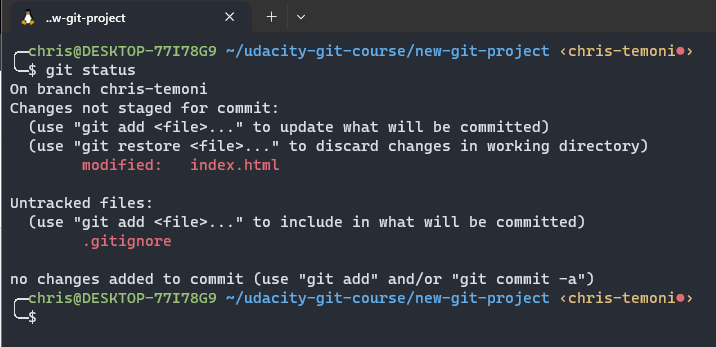
**Command:** code .

**Output:** when executing the code . file it will open up the editor (Vscode).  ****



In the editor navigate to the .gitignore file and add the project.docx line within the file. Save the changes and close the editor

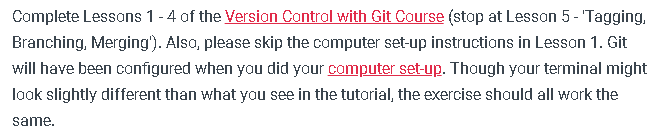
**Run the git status command**

**Command**: git status  
**Output**: the project.docx file is no longer listed as untracked file but the .gitignore file is listed

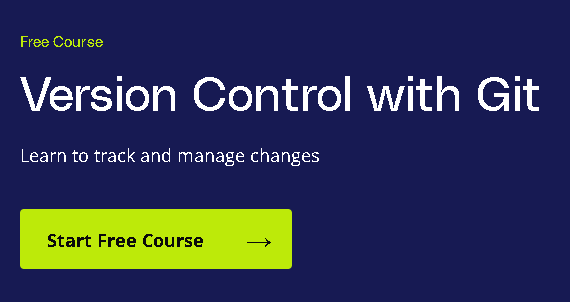
Observations

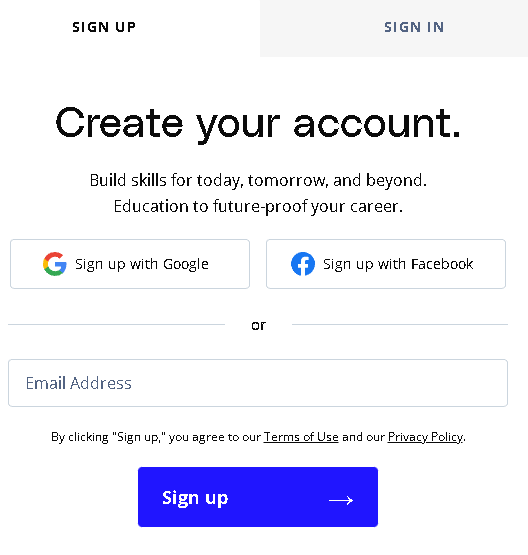
## Version Control with Git – Course

Navigated to [the https://handbook.eda.nz/foundations/curriculum/sprint-1/04-tech/t4-git-install-and-explore](the%20https:/handbook.eda.nz/foundations/curriculum/sprint-1/04-tech/t4-git-install-and-explore) link and scrolled down and clicked the link as indicated in the image (Version Control with Git Course). This link provides a free course to complete lessons 1 – 4 for required task.

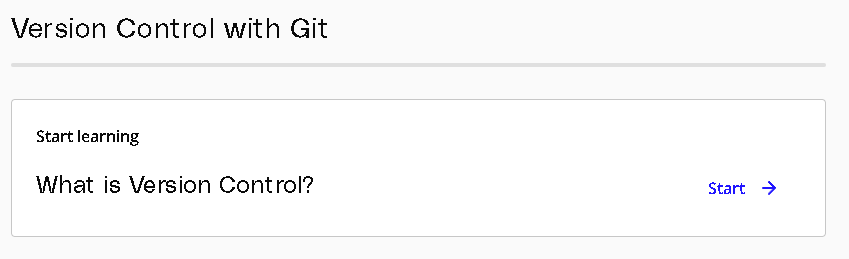


Once linked was clicked Udacity website pops up. Once website popped up I clicked on the Start Free Course link.





Sign up and Sign in window pops up when clicking on the Start Free Course created an account with google. Once account was created, I clicked on sign up link.

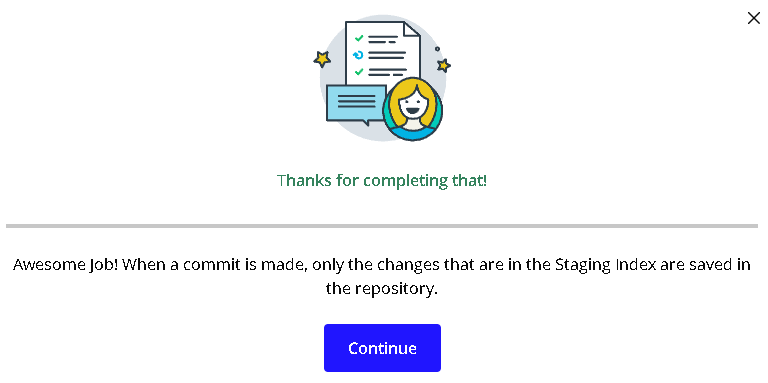


After signing up window pops up with an outline of the course and a link to start learning about version control. Click on the start link to start learning.

## Quiz

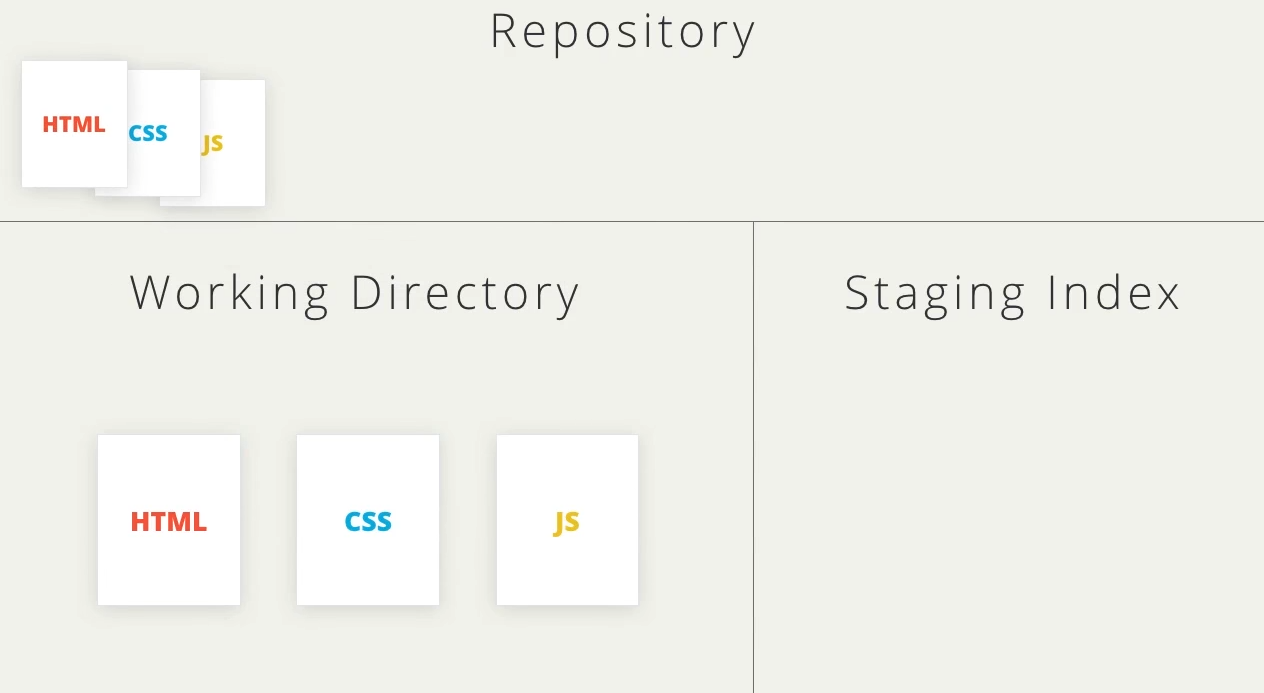


Git and version control terminology quiz



When a commit is made, only the changes that are in the staging index are saved in the repository

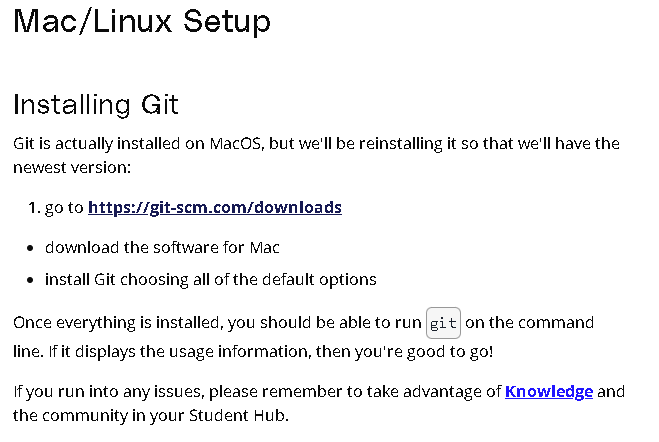
Files that have been committed



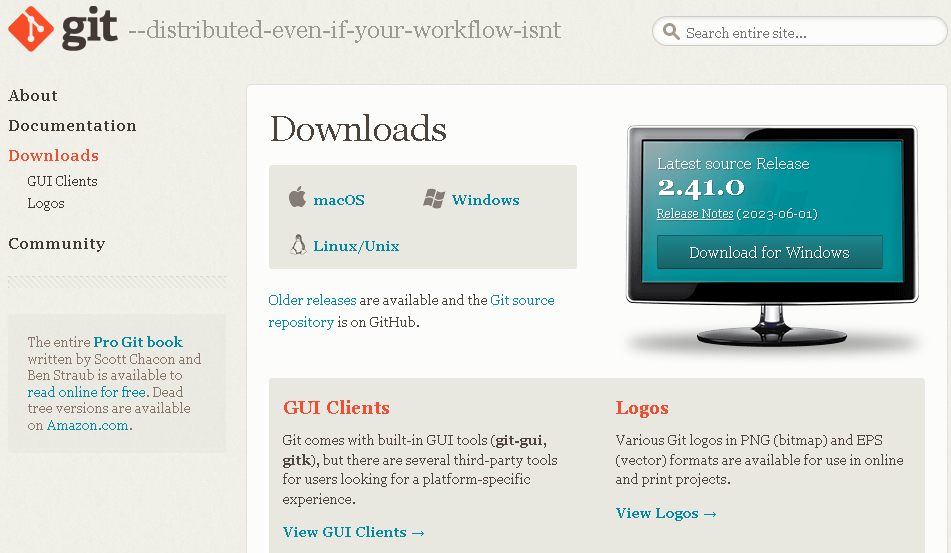
Place files that are about to be committed

Files in working directory have to be moved to staging index in order to be committed

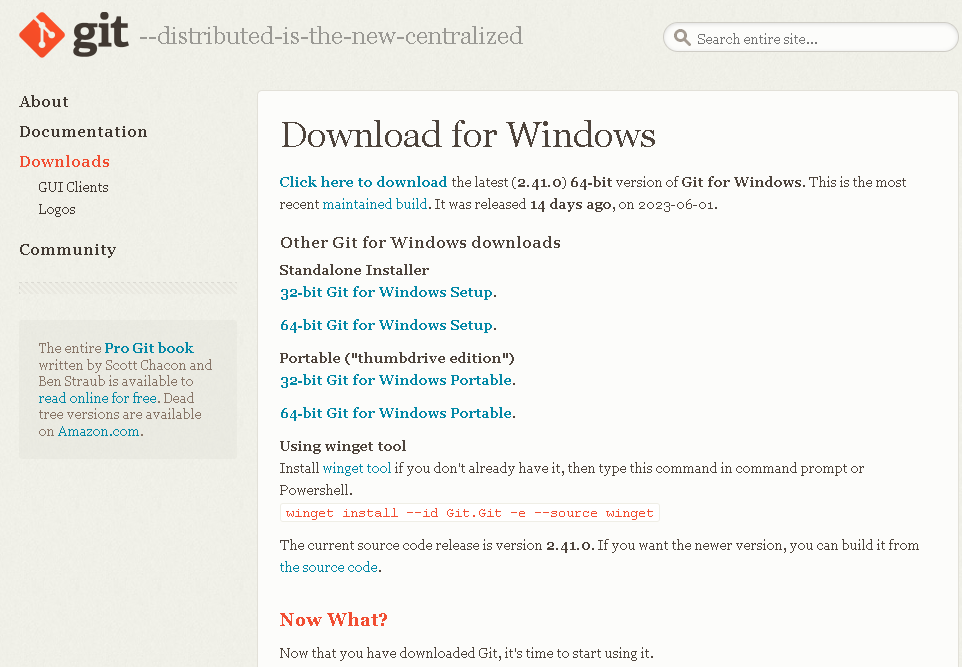
## Install Git



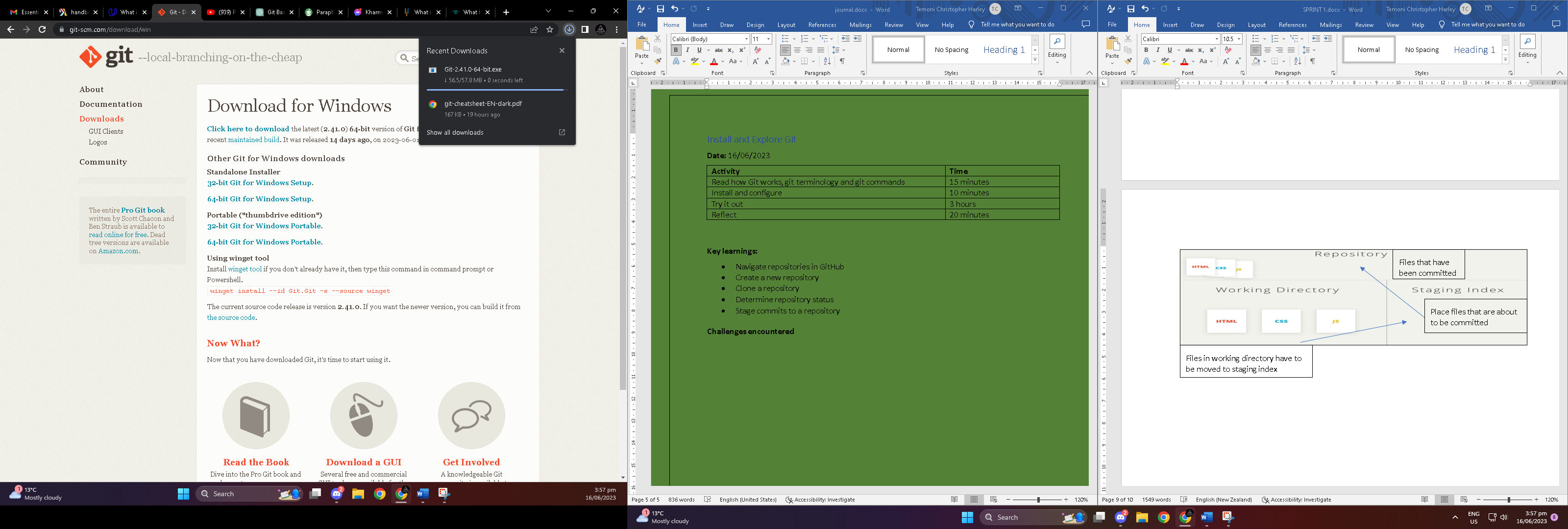
In this part of the free course navigate to the link as displayed in the image



When this window appears click on the windows link



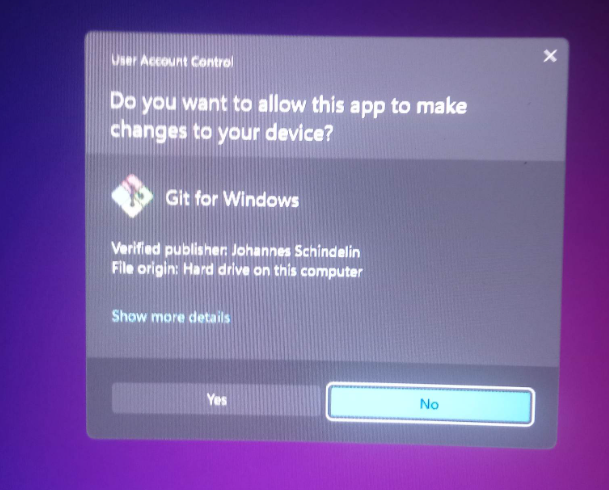
New window opens up click on the first link to start download process



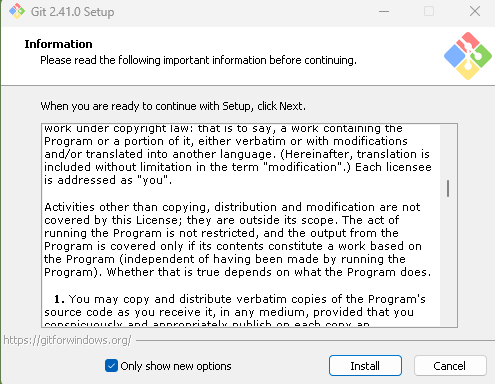
Once you click on the download link above the download process will begin



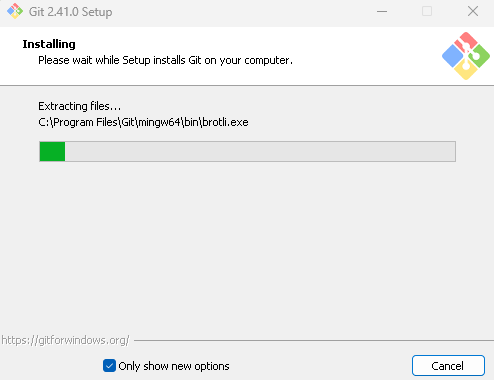
Once download has completed click on the executable file to start the wizard installation process



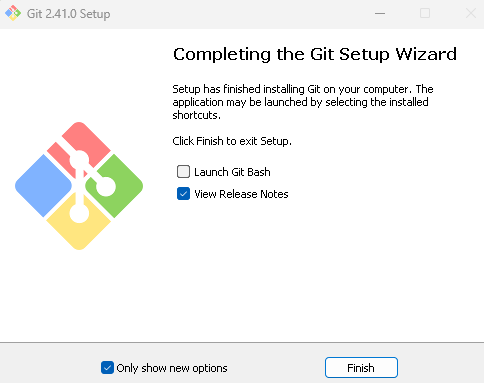
When clicking on the executable file message pops up asking to allow the app to make changes. Click yes



Once you click yes in the above process information window pops up read the following information click on install



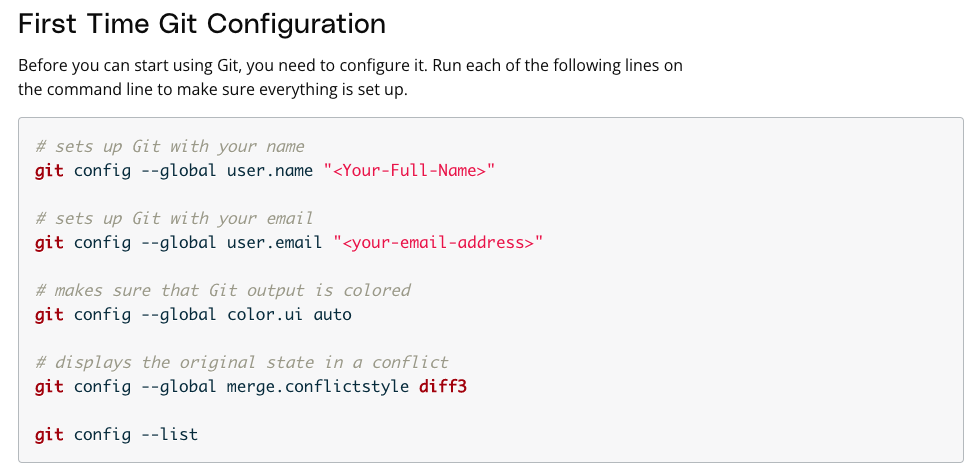
After clicking install the installation process will begin



Installation of git has been completed click on finish

## First time git configuration

The following git commands were executed in the terminal shell.

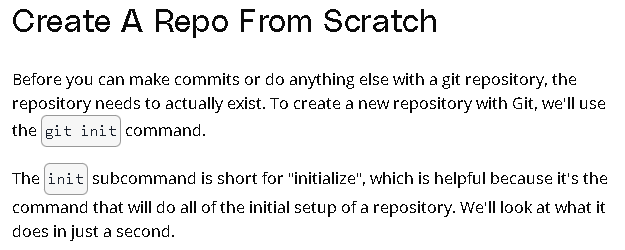




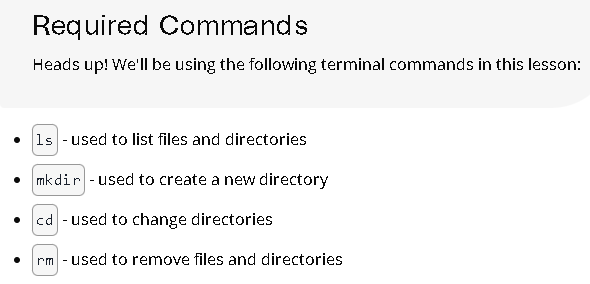




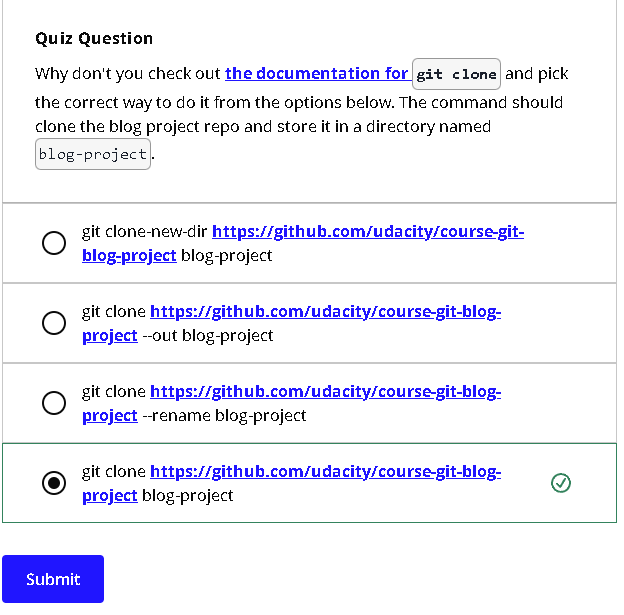
## Create a Repo from scratch



Before creating a repository, I executed the command indicated in the image. (Git init).



The following commands are going to be used for the required task.

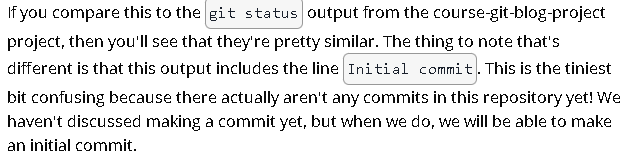


Quiz for git clone

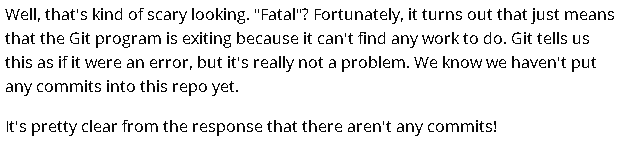
**Git status**

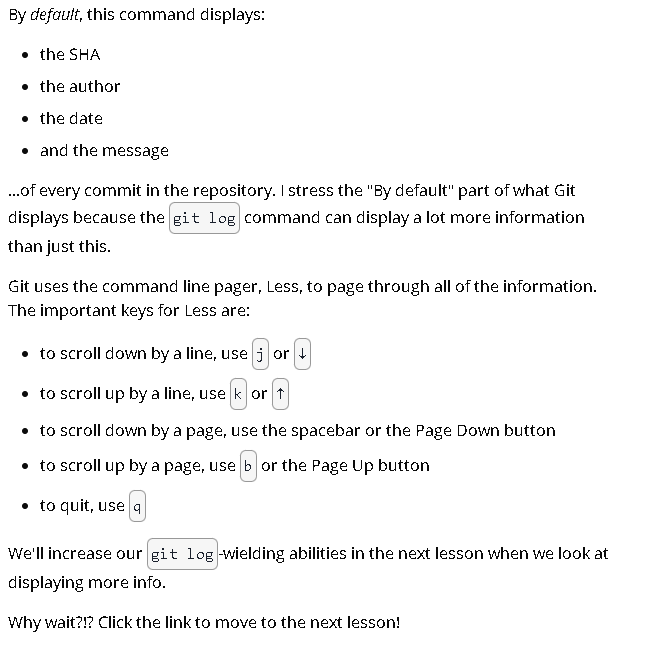


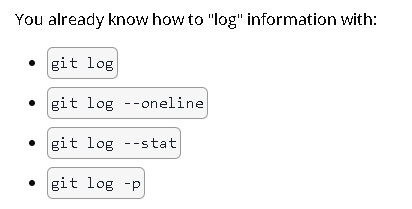
**Git status in new-git-project directory**

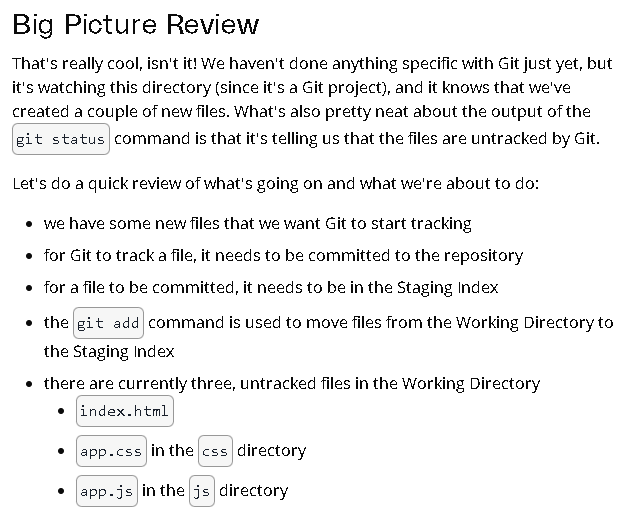
****

**Git log**

****

****

****

**Git add**

**Branch, Pull, Merge**

Activities

**What is a branch**

Using a branch, which is a parallel line of development, you can work on several areas of a project at once. It's similar to making a different copy of your project that you may edit independently of the main branch and other branches.

**Why we use branch**

There are several reasons why we use branch e.g.

* **Parallel Development**
* **Experimentation and Risk-Free Testing**
* **Version Control and History**
* **Collaboration and Code Review**
* **Isolation of Changes**

**What is a pull request**

A pull request is a way to suggest and examine changes made in a repository branch. It frequently appears in workflows for collaborative development, particularly when dealing with shared repositories and team-based projects. Pull requests provide you the opportunity to contribute code modifications and start a conversation about them before they are merged into the main branch.

**Why we use pull request**

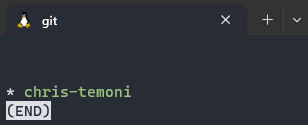
There are several reasons why we use pull request

* **Continuous Integration and Testing**
* **Risk Mitigation**
* **Transparency and Documentation**
* **Collaboration and Discussion**
* **Code Review**

**Use the git branch command**

**Command:** git branch

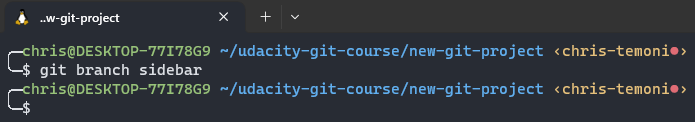
**Output:** output displays a list of branches in a repository

****

**Create a branch named sidebar**

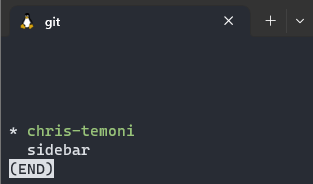
**Command:** git branch sidebar

**Output**: sidebar branch has been created

****

**Check to see if sidebar branch has been made**

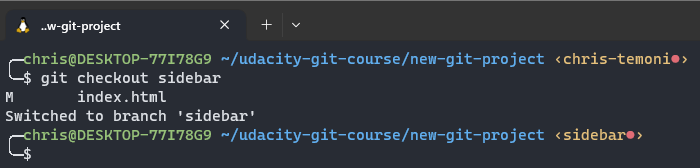
**Command:** git branch sidebar

**Output:** sidebar branch successfully created. Although we have created the sidebar branch, we are still connected to the chris-temoni branch

**Swich to the sidebar branch**

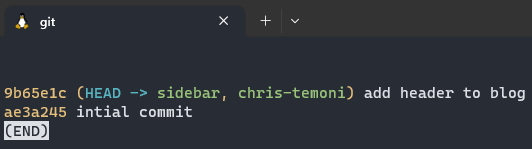
**Command:** git checkout sidebar

**Output**: we have now switched to the sidebar branch

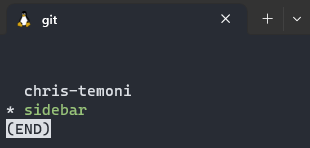
****

**Branches in the log**

**Command:** git log –oneline –decorate  
**Output**: output displays two branches and their respective hashes the head arrow pointing to the sidebar indicates that the sidebar branch I active

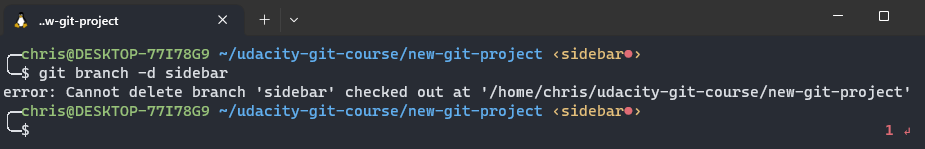
****

**Git branch command**

**Command:** git branch **Output:** output through git branch command shows sidebar has a \* symbol next to and the sidebar text has been highlighted green this indicates that the sidebar branch is active

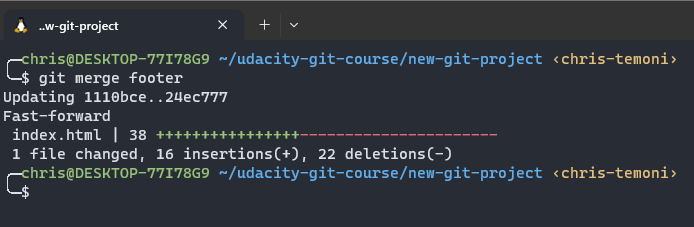
**Attempt to Delete sidebar branch**

**Command:** git branch -d sidebar  
**Output**: cannot delete sidebar branch. You cannot delete a branch that you are currently in you can only delete a branch that you are not in.

****

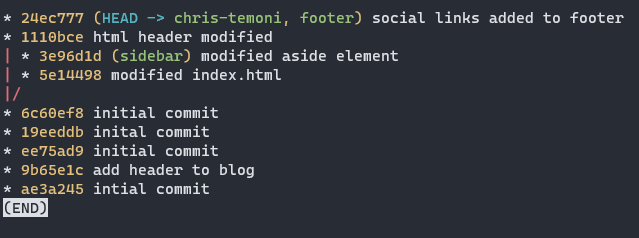
**Merging**

**Command:** git merge footer  
**Output**: we are merging the footer to the chris-temoni branch (master)

****

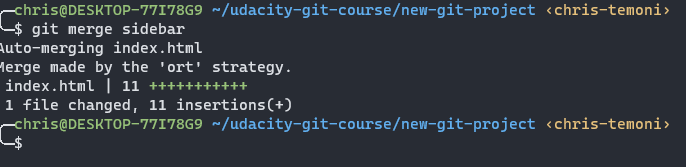
**check to see if footer has merged with chris-temoni branch**

**Command**: git log --oneline --decorate --graph –all  
**Output**: the footer branch has successfully merged into the master branch (chris-temoni)

****

**Merge sidebar branch**

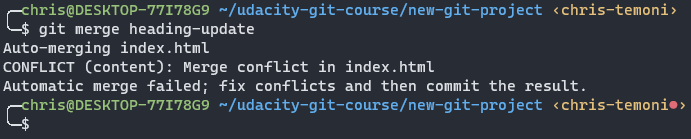
**Command:** git merge sidebar **Output:** sidebar branch has been merged

****

**Forcing a merge conflict**after changing the same line on both the heading-update branch and the master branch (chris-temoni) we are going to try merge both branches

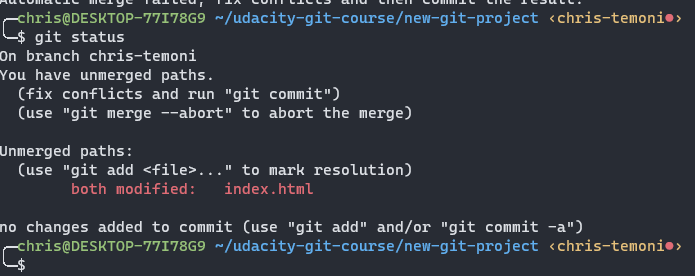
**Merge heading-update and master branch**

**Command:** git merge heading-update  
**Output**: Output displays as expected there has been a conflict error with merging the two branches



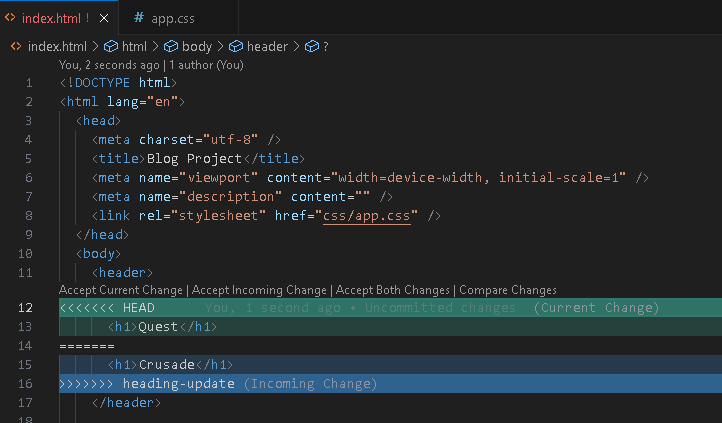
**Git status command**we are going to run a git status command to see where the error is located

**Command: git status  
Output:** output indicates that the error is showing up in the index.html file

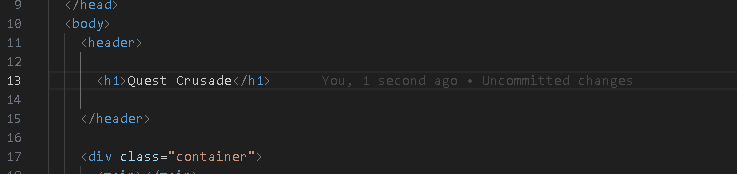


**Go to the code editor and navigate to the index.html file**

**Command:** Code .  
**Output**: code editor displays the conflict issue inside the index file both files had the same modifications on the same line within the index file

****

**remove one of the h1 comments**In the editor remove the heading-update arrows in line 16 and the head arrows in line 12 along with the ==== characters in line 14. Next remove line 15 (crusade) and rename line 13 to quest crusade save changes and close the editor.



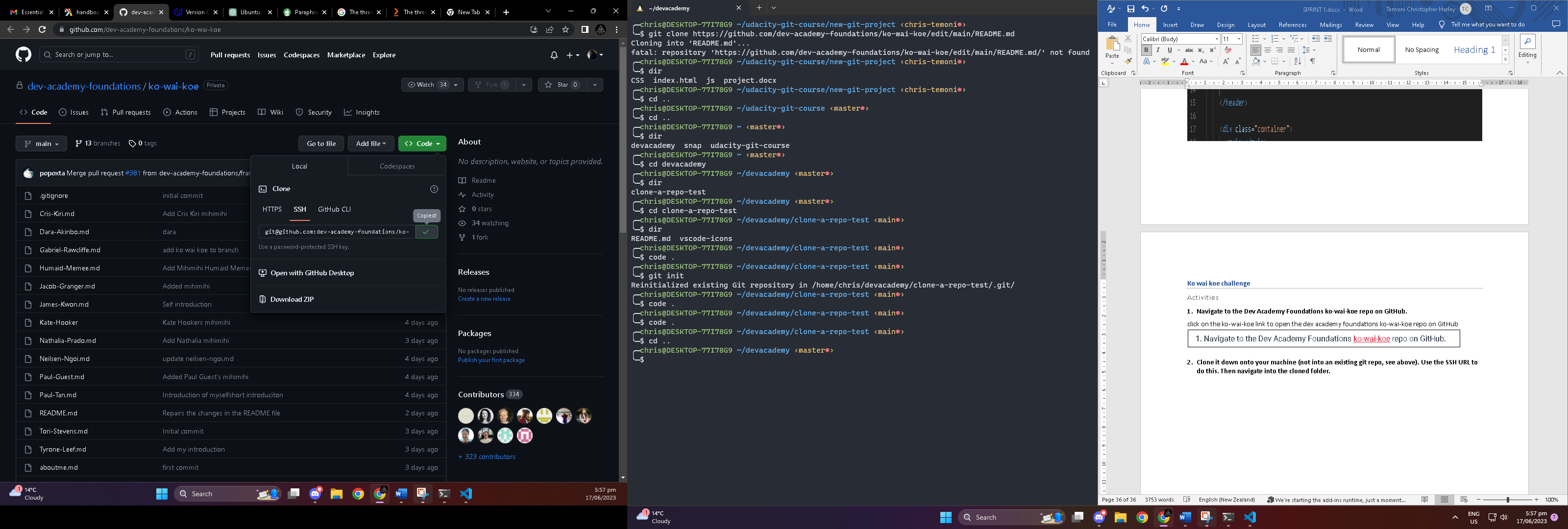
**Ko wai koe challenge**

Activities

1. **Navigate to the Dev Academy Foundations ko-wai-koe repo on GitHub.**

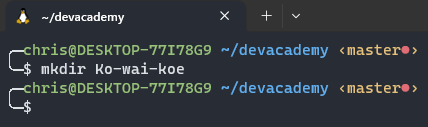
click on the ko-wai-koe link to open the dev academy foundations ko-wai-koe repo on GitHub

1. **Clone it down onto your machine (not into an existing git repo, see above). Use the SSH URL to do this. Then navigate into the cloned folder.**



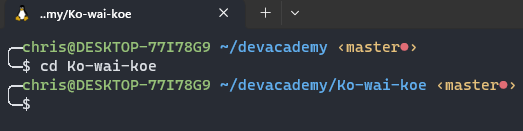
**Navigate to the code drop down arrow then go to the local ribbon. click on the SSH ribbon and copy the URL**

**Navigate to terminal and create Ko-wai-koe directory  
Command:** mkdir Ko-wai-koe  
**Output**: directory created

****

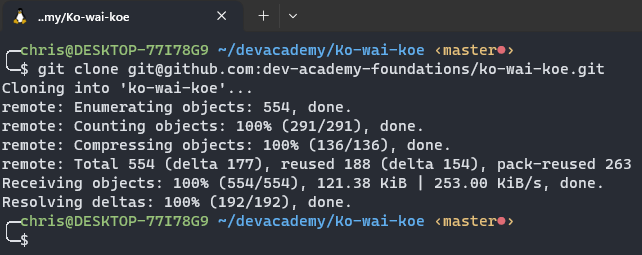
**Change to the new directory**

**Command**: cd Ko-wai-koe **Output:** successfully changed to theKo-wai-koe directory



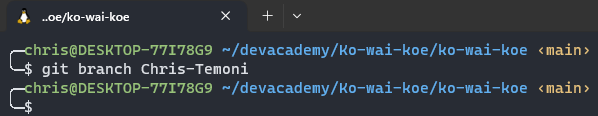
**Clone the SSH copied URL into the**

**Command:** git clone [git@github.com:dev-academy-foundations/ko-wai-koe.git](mailto:git@github.com:dev-academy-foundations/ko-wai-koe.git)  
**Output**: URL cloned successfully



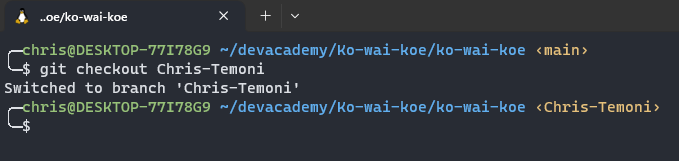
1. **Create a branch using your full-name.**

**Command:** git branch Chris-Temoni  
**Output**: branch created

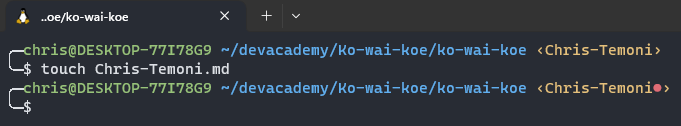
****

1. **Switch (checkout) to the new branch**

**Command:** git checkout Chris-Temoni  
**Output**: checkout successful

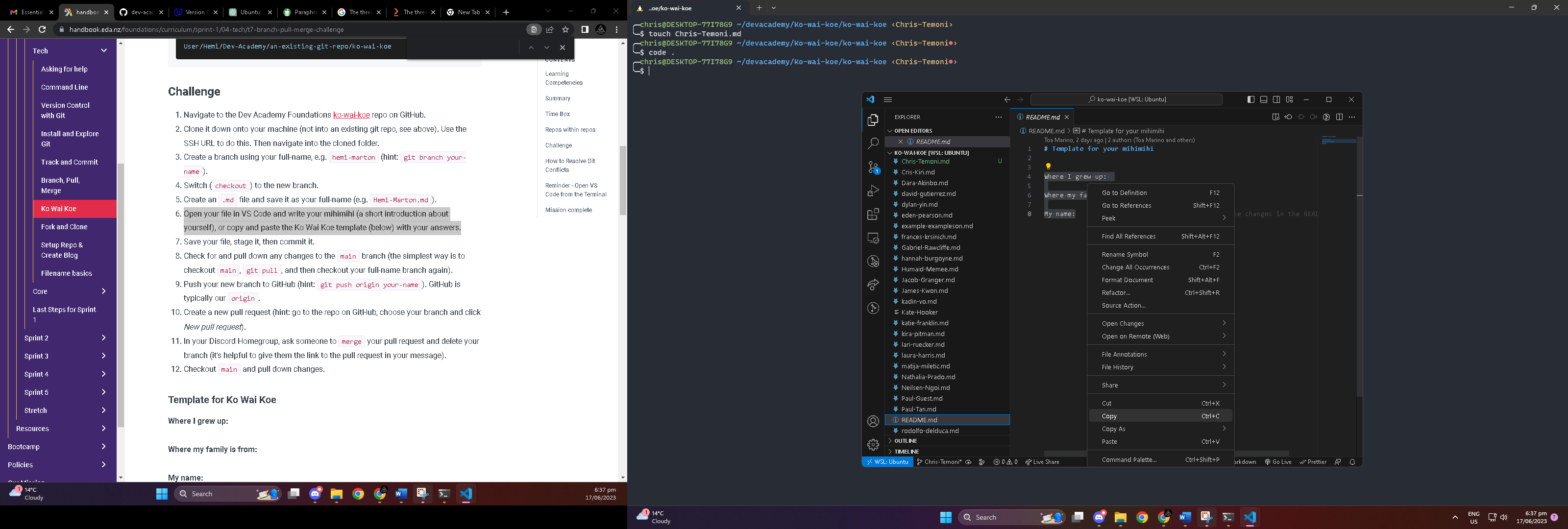


1. **Create an .md file and save it as your full-name.  
   Command:** touch Chris-Temoni.md  
   **Output**: Chris-Temoni.md file has been created

****

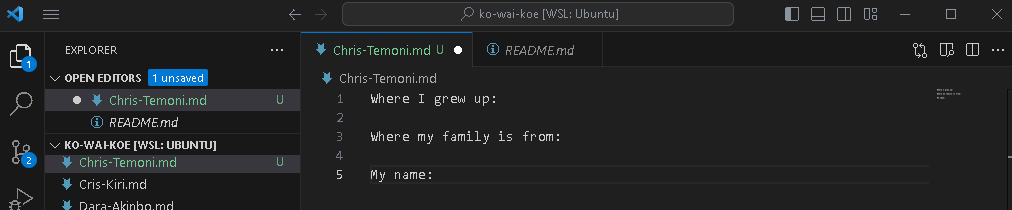
1. **Open your file in VS Code and write your mihimihi (a short introduction about yourself)**

**Command:** Code .  
**Output**: code editor opens up after executing the above code.

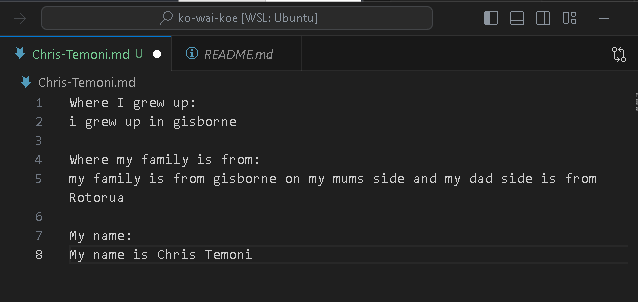


Navigate to the README.md file. Highlight the README text from line 4 to line 8, right click and click on copy

**Step 2**: Navigate to Chris-Temoni.md file and paste the text as indicated in the image



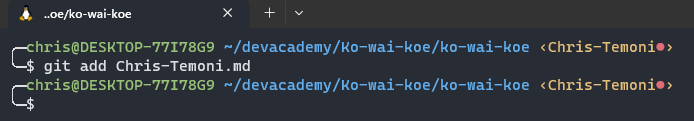
**Fill out the information**

****Once information has been filled out save and close the file editor

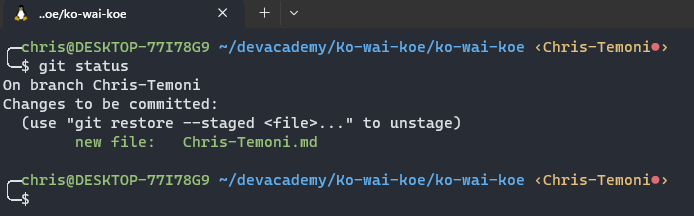
1. **Save your file, stage it, then commit it.**

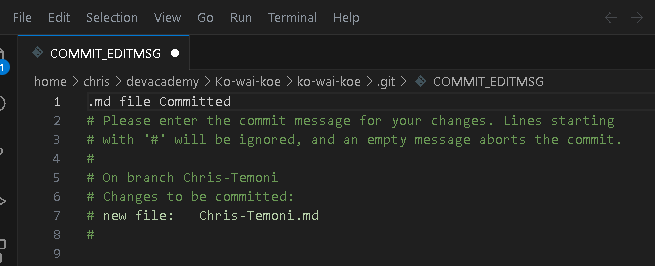
File has been saved

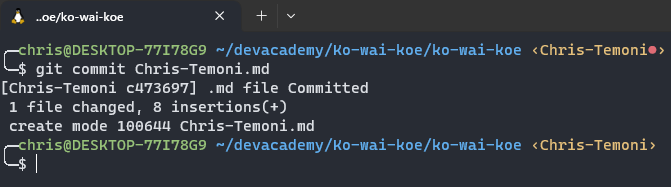
**Stage and commit your Chris-Temoni.md file**

**Stage   
Command:** git add Chris-Temoni.md  
**Output**: File has been staged  


**Check to see if its been staged  
Command:** git status  
**Output**: file successfully staged

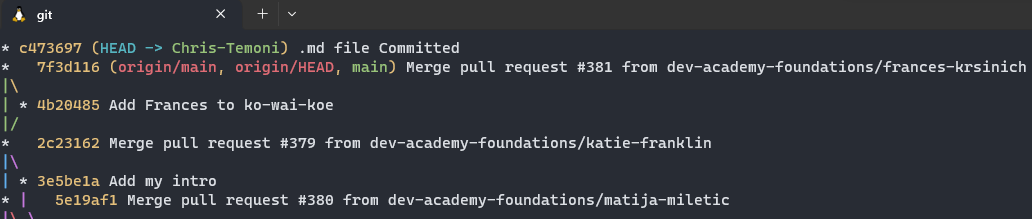


**Commit the file  
Command:** git commit Chris-Temoni.md  
**Output**: code editor pops up after executing above command in line 1 write a message to confirm the commit

**File committed successfully**

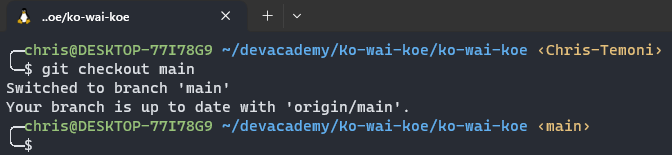
**Check status of the file**

**Command:** git log --oneline --decorate --graph –all  
**Output**: file and branch update information as indicated in the image



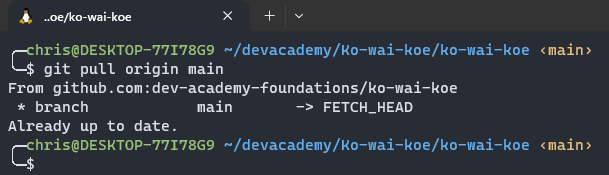
1. **Check for and pull down any changes to the main branch (the simplest way is to checkout main, git pull, and then checkout your full-name branch**

**Command:** git checkout main **Output:** Changed to main

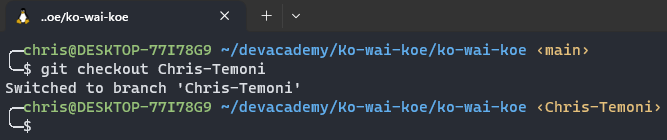


**Pull the latest changes from the remote main branch**

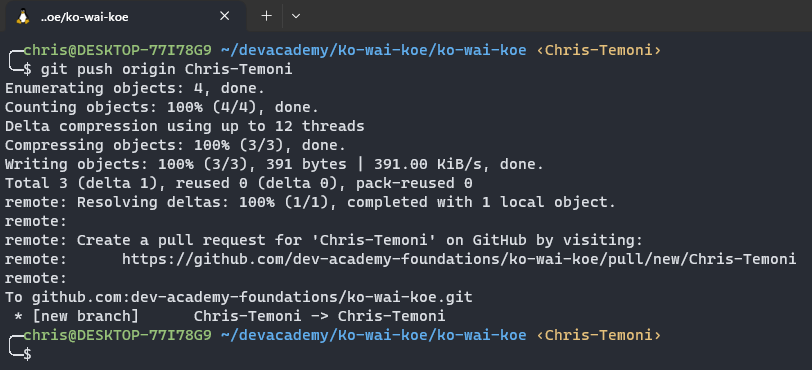
**Command:** git pull origin main  
**Output**: command fetches the latest commits from the main branch and incorporates them into the local main branch

****

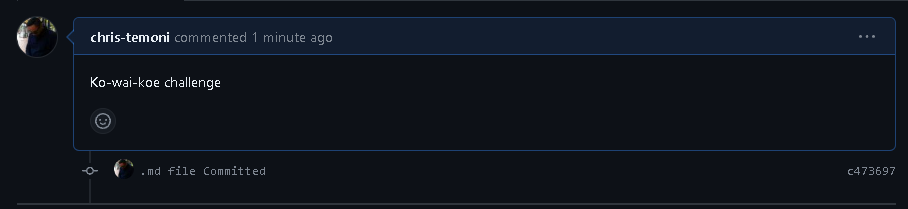
**Switch back to Chris-Temoni Branch  
Command:** git checkout Chris-Temoni  
**Output**: branch changed successfully



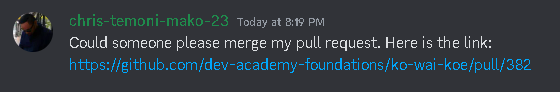
1. **Push your new branch to GitHub (hint: git push origin your-name). GitHub is typically our origin.**

**Command:** git push origin Chris-Temoni  
**Output**: branch pushed successfully

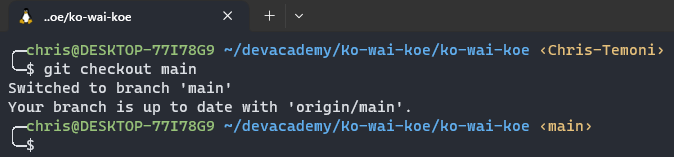
1. **Create a new pull request.**

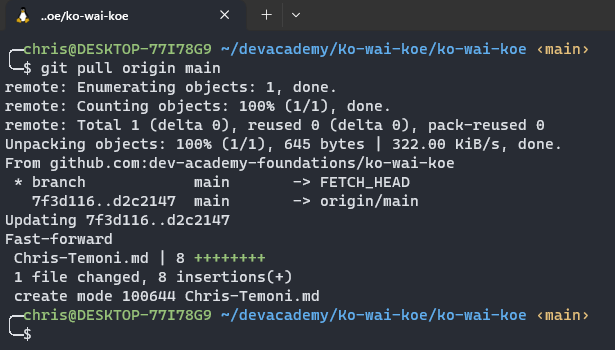
****

1. **In your Discord Homegroup, ask someone to merge your pull request and delete your branch (it's helpful to give them the link to the pull request in your message).**

****

1. **Checkout main and pull-down changes**

**Command:** git checkout main **Output:** switched to main branch is up to day ****

**Command:** git pull origin main **Output:** changes have been pulled

**Fork and Clone**

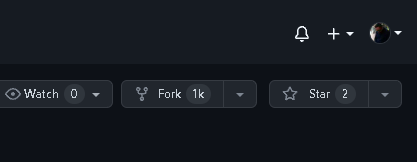
Activities

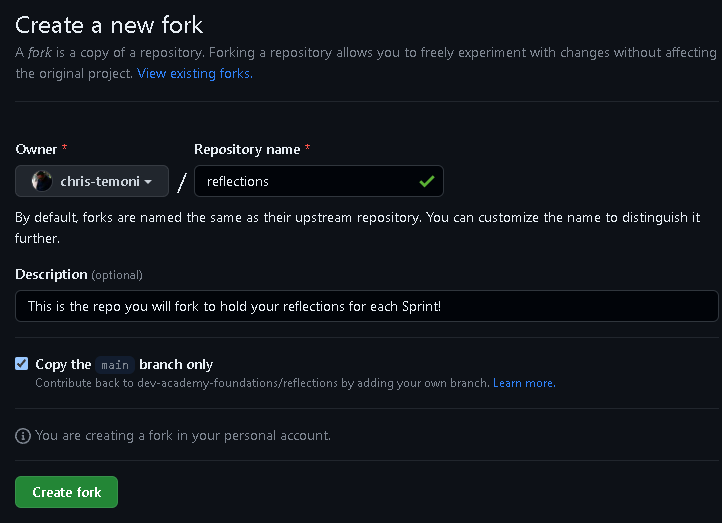
**Step 1: Fork**

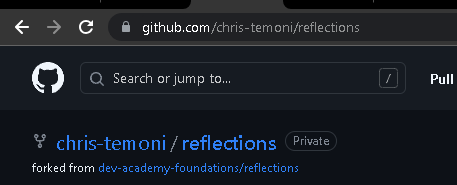
Navigate to the link indicated in the image below



Once you have clicked the link navigate to fork



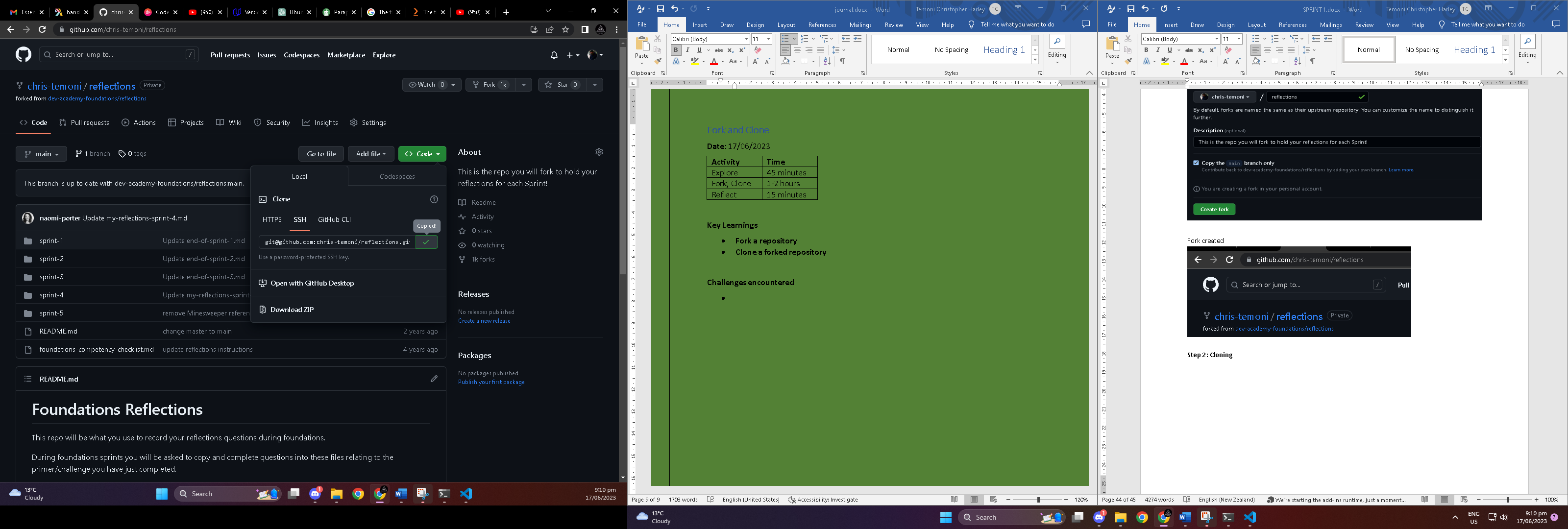
Set owner as yourself leave everything set to default and click create fork

Fork created

**Step 2: Cloning**

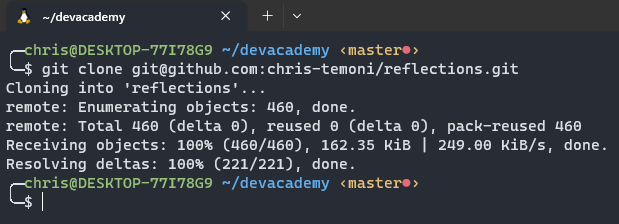
**Copy the SSH link**

In the code drop down button navigate to local ribbon and click on the SSH ribbon and copy the link as indicated in the image



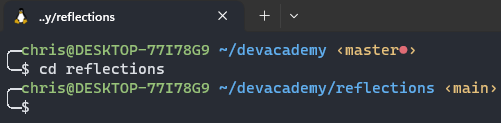
**Clone the copied link into your terminal in the devacademy directory**

**Command**: git clone [git@github.com:chris-temoni/reflections.git](mailto:git@github.com:chris-temoni/reflections.git)  
**Output**: clone successful



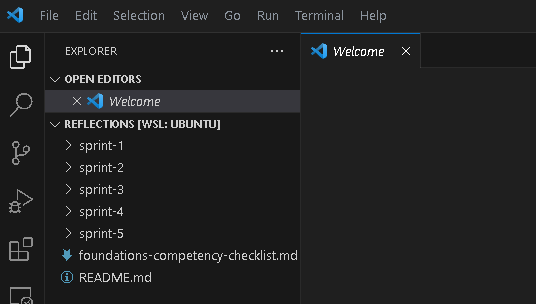
**Step 3: open in Code editor**

**Change to the reflection’s directory**  
**Command**: cd reflections  
**Output**: changed directory successfully



**Open code editor (Vscode)**

**Command**: code .  
**Output**: above command opens up the code editor



**Introduce yourself in Discord (#Foundations)**

