GitHub is a platform for version control and collaboration, enabling multiple developers to work on and manage code projects together, track changes, and coordinate contributions efficiently.

The purpose of using GitHub in our case is to create a central location to house code and track processes.

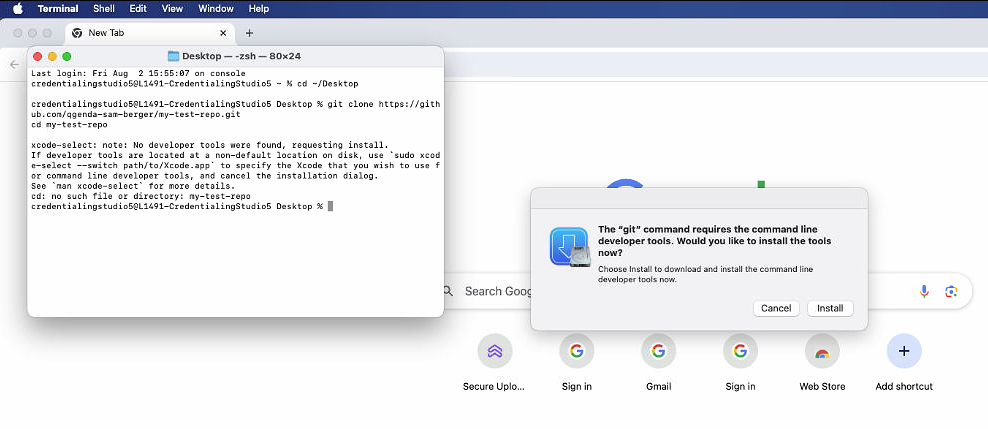
**Creating a Personal Access Token on GitHub**

* Go to GitHub and click your profile on the top right
* Scroll down to Settings
* Once there, scroll down to Developer settings, all the way at the bottom
* Open the Personal access tokens dropdown and choose Fine-grained tokens
* Click “Generate new token”
* Name it something appropriate. Mine is called my-PAT
* Leave the Expiration length where it is
* A description is not necessary but you can add a generic one if you like
* Under repository access, choose “All repositories”
* Under permissions, and under repository permissions, for the following options, change the access to “Read and Write”
  + Actions
  + Administration
  + Contents
  + Deployments
  + Issues
  + Pull requests
  + Note that Metadata is mandatory and can remain Read-only
  + Under permissions and under account permissions, change email addresses to read and write
* Choose generate token

**Installing the “command line developer tools”**  
In order to use GitHub on the virtual machines, you need to make sure the command line developer tools are installed.

To see if you have them installed, you can open the Terminal app (open the launchpad and search for Terminal on the top).

If you don’t have it, the following prompt will open.



1. Click install when the prompt shows up

**Interacting with GitHub using the Terminal app on Mac computers**

Remember to access Terminal, as that is how we interact with GitHub on our machines.

1. Open the launchpad and search for Terminal
2. Point Terminal toward the directory you want to access
   1. In this case, on Studio 3, our code lives in Desktop so we point our Terminal toward Desktop by typing “cd ~/Desktop”
3. Clone the repository you want to write to
   1. In this case, we want to access the test repo so we type “git clone https://github.com/qgenda-sam-berger/my-test-repo.git
   2. cd my-test-repo”
4. GitHub will ask for your username
   1. In this case, I would type qgenda-sam-berger as that is my username but you would type your own
   2. 
5. GitHub will then ask for your password
   1. DO NOT INPUT YOUR GITHUB PASSWORD
   2. Instead, create and use the personal access token (PAT)
6. Point the Terminal toward the directory you’re working in
   1. For instance, I created test.sql and test2.sql in GW copy so I am using this command:
      1. cd /Users/credentialingstudio4/Downloads/Slipstream/Code/SQL/SQL\ v2.2\ -\ GW\ copy/
   2. Once there, add your code to Github by using the “git add” command
      1. In my example, I am typing “git add test2.sql”
   3. Type a simple message to explain what you’re doing
      1. In my example I am typing “git commit -m "Add test2.sql file" as I am simply adding test2.sql to GitHub as another example
   4. Once you enter the commit message, push the change to GitHub by typing the following:
      1. “git push origin main”
   5. I then like to refresh GitHub and make sure the new item shows up

**Common Git commands we will use**

* git clone <repo-url>: Clone a remote repository to your local machine.
  + You should only need to clone the repo once
* cd <file path>
* You may need to move the file to the correct path
  + You will get an error telling you, you need to move it
  + To move it, use this command: mv /path/to/source /path/to/destination
* git add <file>: Stage changes (add files) for the next commit.
* git commit -m "message": Commit staged changes with a message.
* git push origin main

**Common Git commands and their meanings**

* git init: Initialize a new Git repository.
* git clone <repo-url>: Clone a remote repository to your local machine.
* git add <file>: Stage changes (add files) for the next commit.
* git commit -m "message": Commit staged changes with a message.
* git status: Show the status of your working directory and staged changes.
* git push origin <branch>: Push commits from your local branch to a remote repository.
* git pull origin <branch>: Fetch and merge changes from a remote branch into your local branch.
* git checkout <branch>: Switch to another branch.
* git branch: List, create, or delete branches.
* git merge <branch>: Merge another branch into your current branch.
* git log: View commit history.