Lab 1 – Setup Git and GitHub

**Part 1 – Accessing Git Program**

1. A screenshot of a computer

   Description automatically generated with medium confidenceUsing the GoToMyPC provided Virtual Machine or an already installed version of Git on your computer. Type into the Windows Search command ‘Git Bash’ You should have an application that looks like the below Screenshot
2. Text

   Description automatically generatedOpen this application and it should open up a terminal window that looks like the below
3. Text

   Description automatically generatedThis is your Git Bash terminal that replicates a Linux terminal inside of your windows machine to use Git with. When using a terminal you type where the $ is and when you have finished your command press enter. In the terminal type in ls and press enter like below
4. This command shows all files in this current location and is simply printing the contents of this current location (which is your home user account)
5. Graphical user interface, text, application

   Description automatically generatedGraphical user interface, application

   Description automatically generatedUsing File Explorer create a new directory in your documents you wish to work in, in this folder right click in the body and select ‘Git Bash Here’
6. This will open up a Git Bash terminal in this location, as shown by the yellow text in the Terminal window, the following are basic linux commands which will be used throughout the Git course

|  |  |
| --- | --- |
| ls | Lists contents of this directory |
| cd <folder> | Changes to that folder from this location |
| cd .. | Goes back a folder, if in /test/folder1 you would go back to /test |
| touch <file> | Makes a new file of the file name you pass in |
| mkdir <name> | Makes a new folder of that name |

Using these commands do the following:

* Create a file called file.txt
* List all of the files in this location
* Create a new directory called “test”
* Navigate to this directory and make two files called file1.js and file2.exe
* Go back to the original folder

**Stretch goal** – Use the terminal to Delete the folder “test” and all of its content

**Part 2 – Accessing GitHub and connecting it to our Git**

1. Go to this website <https://github.com/> and follow the steps to create a Git Hub Account, preferably using a personal email so you always have access to this GitHub Account
2. Graphical user interface, text, application

   Description automatically generatedOn the GitHub GUI click in the Top Right corner Drop Down Arrow next to the Plus and click ‘New Repository’
3. On the new page give your Repository a name, don’t change any of the other settings and click ‘Create Repository’
4. Graphical user interface, text, application, email

   Description automatically generatedThe next page should look like the below (with the GitHub URL being the name of your repo  
     
   Copy the command block that starts with ‘echo “# repo name..’ and ends with ‘git push -u origin main’ and put this into a new notepad for the time being.
5. Open up a GitBash terminal and enter the following (replacing email and username with your GitHub User name and email)

git config --global user.name [USERNAME]

git config --global user.email [EMAIL]

1. In the same GitBash terminal copy the first line from the notepad and paste it into GitBash (You cannot ctrl+v this, you have to rightclick or press Shift + ins), do this for all lines stopping at the Git Push command
2. Copy the Git Push command and enter it, when you send this to GitHub it should open up by default a program called ‘GitHub Credential Manager’ depending on how Git has been installed. This program will ask for your username and password to authenticate you. If this does not open and the Git bash tells you you are not permitted you will need to authenticate via Personal Access Token <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token> or via SSH <https://docs.github.com/en/authentication/connecting-to-github-with-ssh>  
     
   If either of these are true let your trainer know to help authenticate and set you up

Lab 2 – Staging Area

Prerequisites: GitHub Account, machine with Git Installed and Authenticated, Git Repo on GitHub and local we can work with

In your local Git Repo accessible via Git Bash do the following commands to commit and push files:

touch newfile.txt

git add newfile.txt

git commit -m "added newfile"

git push

The first command creates our file, the following three git commands Add a file to staging area, Commits the contents of the staging area and pushes the commits to your GitHub Repo.   
  
Using this concept do the following:

* Create file1.js, file2.txt, file3.java and push them all up to GitHub in two separate commits (hint, one commit will require two files in staging area)
* Delete file1.js and commit this change, push this up to GitHub
* Rename file2.txt to fileTwo.md and push this to GitHub

Stretch goal – Create a directory called /myDir and add 2 files to this location. Push both of those files and the directory to the staging area

Lab 3 – Branching

Prerequisites: GitHub Account, machine with Git Installed and Authenticated, Git Repo on GitHub and local we can work with

Using the Git Branch commands setup the following environment:

* Dev Branch containing the file dev.txt off of main branch
* Feature1 branch containing feature1.md off of Dev branch, repeat this so you have 3 Feature branches
* Push up all of your branches and files to GitHub, on GitHub the feature branches should ONLY CONTAIN THEIR CORRESPONDING FILES
* Once the branches are pushed to GitHub, delete the feature branches locally

**! IMPORTANT MAKE SURE YOU SWITCH BACK TO DEV BETWEEN FEATURE BRANCHES !**

Lab 4 – Logging and Reverting

Prerequisites: GitHub Account, machine with Git Installed and Authenticated, Git Repo on GitHub and local we can work with

Using the Logging and reverting commands do the following processes:

* Show all commits that have been made in this repo
* Show all commits in an easier to read format
* Show all commits and branch graphs
* Create a commit that contains a file called password.txt and Revert back to before you committed it (it should stay in the log)
* Create a commit containing SUPERSECRETPASSWORD and Reset to before you sent the commit (it should disappear from the log)

Lab 5 – Pull Requests

**Part 1 – Merging your own changes into your own repo**

In GitHub on your personal repo where you have multiple branches follow the following steps:

1. Click on the tab ‘Pull Requests’ and from here click on ‘New Pull Request’

Graphical user interface, application

Description automatically generated

1. Graphical user interface, text, application, email

   Description automatically generatedFrom here set the Base to be your branch you want to merge changes into (Dev) and set the Compare to the branch you want to merge the changes from (Feature1). Click Create Pull Request
2. The next page gives you a title to set and a description, feel free to enter what you are merging and a quick description if you want, then click ‘Create Pull Request’
3. Graphical user interface, application

   Description automatically generatedOn the next page click ‘Files Changed’ to see the files you have added, you can click ‘Review Changes’ and approve the file as well as giving a comment. After this click back to ‘Conversation’ and scroll down and click ‘Merge Pull Request’

If you followed the steps you merged code from one branch to another within GitHub, you can see these new changes if you pull the changes from your local Git Bash.

**Part 2 – Merging other peoples code and Reviewing**

Within your Breakout groups (2 – 3 people) you are aiming to be invited as collaborators to each others repos, Pushing a change to the repo (either through Git Bash or the GitHub Web Editor), making a pull request and then reviewing the changes.   
  
The first step is to invite Collaborators, this can be found through Repo > Settings > Collaborators and entering their Git Usernames.   
  
Once you are invited to a Repo, you should make a new branch with your name and a file called Hello.txt that has a simple text message inside. You should follow the steps of Part 1 with the only difference being you set the code owner to review your code.

On the following page the code reviewer should suggest a suggestion or approve the code before the code writer merges the pull request