Lab 1 –An introduction to CST8359

# Description

This lab is designed to prepare you for CST8359.

# Estimated Time

This lab will take an estimated 4 hours to complete.

# Deliverable

* Push your code to GitHub and submit the GitHub link to Brightspace.
* Copy and paste the git logs to Brightspace.

# Step 1: Install Visual Studio

Before everything, we prepare our environment for the rest of the course, so let us install Visual Studio.

1. Use a web browser and navigate <https://visualstudio.microsoft.com/free-developer-offers/>
2. Download the community edition of Visual Studio 2022
3. Once downloaded, install Visual Studio. Note that you will want to ensure that you have selected the following features:
   * ASP.NET and Web Development
   * Azure Development
   * .NET Desktop Development
   * Data Storage and Processing
   * Visual Studio Extension Development

Also note, you are free to install whichever tools you would like. I highly encourage looking into the Universal Windows App Development Tools (Windows 10 apps), as well as Mobile Development, and Cross Platform Development.

1. Progress with installation, this could take a while.
   * Maybe continue with the rest of this lab while waiting!

# Step 2: Install git.

We also need Git. We will use git to submit assignments in this course. So, let’s learn how to work with it.

1. Follow the instructions [here](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git) to install git!
   * While there are some great git GUIs (graphical user interfaces), I think it's easier to learn git using git-specific commands first and then to try out a git GUI once you're more comfortable with the command. 95% of online git resources and discussions will also be for the command- line interface.

# Step 3: Create a git account.

If you already have a GitHub account, feel free to use it. Otherwise, create a GitHub account [here](https://github.com/join).

# Step 4: Create local Git Repo

When creating a new project on your local machine using git, you'll first create a new [repository](https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository) (or often, 'repo', for short).

1. Open git bash or any terminal.
2. cd to your favorite folder
3. create a new folder: mkdir myproject
4. cd myproject
5. run git init to create a repo: git init

# Step 5: Add a file to the repo

1. Using any text editor of your choice, add a file to the myproject directory. The file should be named as your Algonquin email address. Like afrasia.txt for [afrasia@algonquincollege.com](mailto:afrasia@algonquincollege.com).
2. Check the git status to verify your file is there: git status
   1. Note that your file is untracked by git at this point
3. Repeat the steps above to create a file called: .gitignore
4. Repeat the steps above to create a file called: cst8359.txt

# Step 6: Check the status of git

Check the status of git. What differences do you compared to the previous step?

# Step 7: ignore some file

1. Using your text editor, modify the file .gitignore and add the following line in it:
   1. \*\*/cst8359.txt
2. Check the status of git. What differences do you compared to the previous step?
3. .gitignore file orders git to ignore specific file from tracking, so changes to those files will not be noticed by git. This is very important because you need to keep some files out of git’s hand for security, cost effectiveness, and general tidiness.

# Step 8: Add file to the staging environment

1. Use `git add .` command to add the file you just created to the staging environment
2. Check the status of git. What differences do you compared to the previous step?

# Step 7: Create a commit

It's time to create your first commit!

1. Run the commit command to commit the files that you already staged to git
   1. git commit -m “your commit message”
2. Check the status of git. What differences do you compared to the previous step?

# Step 8: Create a new private repository on GitHub

Check the following screenshot and create a similar repo on GitHub.

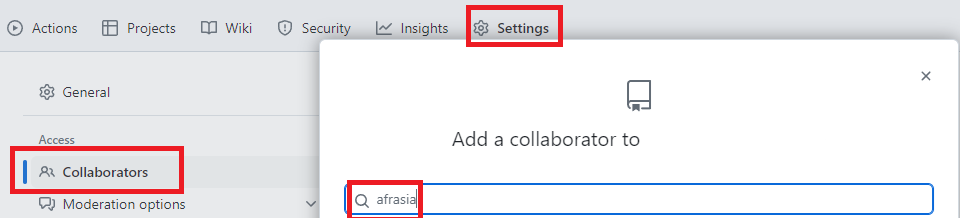
A screenshot of a computer

Description automatically generated

When you're done filling out the information, press the 'Create repository' button to make your new repo. In the next window, GitHub will provide you some commands to push your local repository to GitHub. Locate the command that starts with `git remote add origin`, copy and paste it to your terminal, and hit enter. This will link your local repo to remote GitHub repo.

You can then push your local repo to remote using: git push -u origin main

Notice that you are creating a private repository. Private repositories are only available to you and people you invite as collaborators. To get your assignments marked, you need to add `afrasia` as a collaborator.



# Step 8: Create a new branch

Now that you've pushed your main branch, let's try something a little more advanced.

1. Using the command below, create your first branch called lab1
   1. git checkout -b lab1
2. check what branch you are on: git branch

# Step 10: make changes, stage, commit, and push your branch

1. using your text editor, add a file called <your student number>.txt, like 123456.txt
2. stage your change: git add 123456.txt
3. commit your change: git commit -m “<your message>”
4. push your changes: git push –set-upstream lab1
5. check your GitHub repo, you now have two branches:



1. switch back to your main branch: git checkout main
2. notice that the file 123456.txt disappeared

# Step 11: Create a merge (pull) request (MR/PR)

A pull request (or PR) is a way to alert a repo's owners that you want to make some changes to their code. It allows them to review the code and make sure it looks good before putting your changes on the primary branch.

1. Click on `branches`
2. On the right side of `lab1` branch, click on …
3. Click on `New Pull Request`
4. Click on `Create Pull request`
5. At this point the changes from `lab1` branch are moving to get merged to the `main` branch
6. After reviewing the changes, merge the PR by clicking on `Merge pull request`

# Step 12: Get the changes back to your local repo

1. Notice that the file 123456.txt exists on the remote repo, but it does not show up locally on your device. To get the most recent changes that you or others have merged on GitHub, use the `git pull origin <branch name>` command. In most cases, this can be shortened to “git pull”. Pull the changes from the main branch by: git pull
2. Notice that 123456.txtx appears locally now

# Step 13: check the history of changes

Use the `git log` command to check the history of changes and commits that went on in your main branch. You need to submit this log to Brightspace.