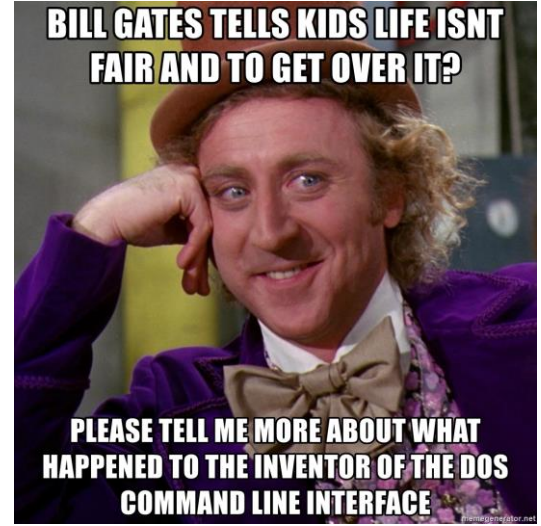


Module 1-1

Command Line Shell & Version Control

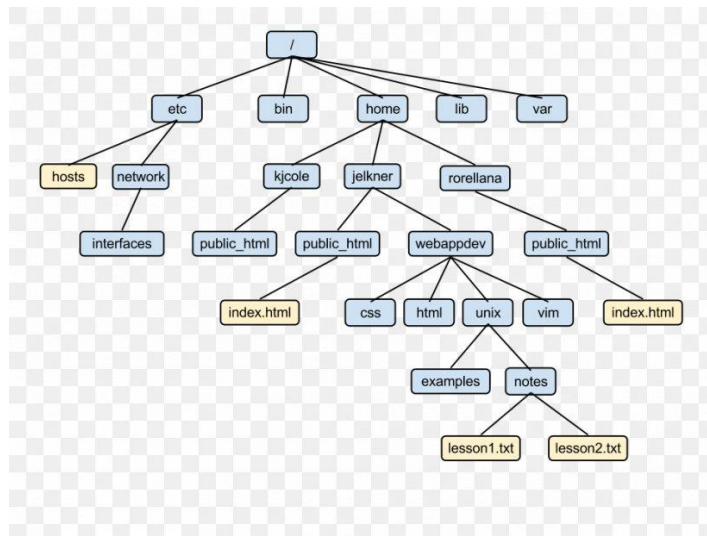


Objectives

- Navigate files using the UI of their laptops
- Find and open a command line application
- Have your repository set up on their laptops
- Open Visual Studio Code as a text editor
- Understand that there is a basic command line on their computers and how to use it
- Understand pathing and hierarchical, parent-child structures
- Know the cd, ls, and pwd commands and how to use them
- Understand what source control is
- Have an understanding of what git is and what the workflow of it will be in the class

File System

- Method for organizing and storing files
 - Organized into tree structures
 - Drives contain folders and folders contain files
- Like a filing cabinet
- Files contain the data we want
 - Documents, spreadsheets, etc.
- Folders hold files and other folders
 - All files exist in some folder in the file system



What is a Command Line Shell?

- A shell is the means by which the user interacts with the computer.
 - Shells can be in the form of a graphical user interface (i.e. Windows, MacOS) – GUI or UI
 - Shells can be in the form of a command line, where users type in commands.
- Information Technology professionals should be familiar with **command line shells**.
- In this class we will be using GitBash, which allows for UNIX commands from a windows workstation.

Command Line Commands: Moving Around

- Data in your workstation are organized into files and folders.
- The main command to move around folder is **cd**. There are several variations of these:
 - **cd ~** : Returns you to your home directory.
 - **cd <directory name>** : Takes you to a specified directory i.e. cd workspace takes you to a folder called workspace
 - **cd ..** : Takes you one level up.
 - **cd /** : Takes you to the root
- You can always see what directory you're in by typing **pwd**.
- The **ls** command lists all the files in the current directory.
- The **ls -al** command will list all the files, including any hidden ones.

Let's Try this!

Moving Around: Absolute Path

- When you used the `pwd` command, the output would have looked something like this:

```
Student@DELL-JAVA MINGW64 ~/workspace  
$ pwd  
/c/Users/Student/workspace
```

Recall that `pwd` displays the current directory. Note that the response from this command is an absolute path since it starts with a slash (/).

Moving Around: Relative Path

- A relative path is differentiated from the absolute path by the absence of the initial slash:
 - **cd /c/Users/Student/workspace** uses an absolute path to get me to the workspace folder.
 - Alternatively, if I were already in my respective user folder (Student), typing **cd workspace** uses a relative path to get me to the workspace folder.

Moving Around: The Tilde (~)

- The tilde (~) is a special symbol used to denote the home directory. For all of your workstations this has been set to: `/c/users/Student`

```
Student@DELL-JAVA MINGW64 ~/workspace
```

```
$ cd ~/workspace
```

Therefore, the above command will take you to: `/c/Users/Student/workspace/`

Moving Around: Making Directories

- To create a directory we use the **mkdir <filename>** command.

Command Line Commands: Copying

- To copy a file from 1 directory to another: `cp <source> <destination>`

```
Student@DELL-JAVA MINGW64 ~
```

```
$ cp ~/testdir/file.txt ~/othertestdir
```

- To move a file from 1 directory to another: `mv <source> <destination>`

```
Student@DELL-JAVA MINGW64 ~
```

```
$ mv ~/othertestdir/file.txt ~/testdir/
```

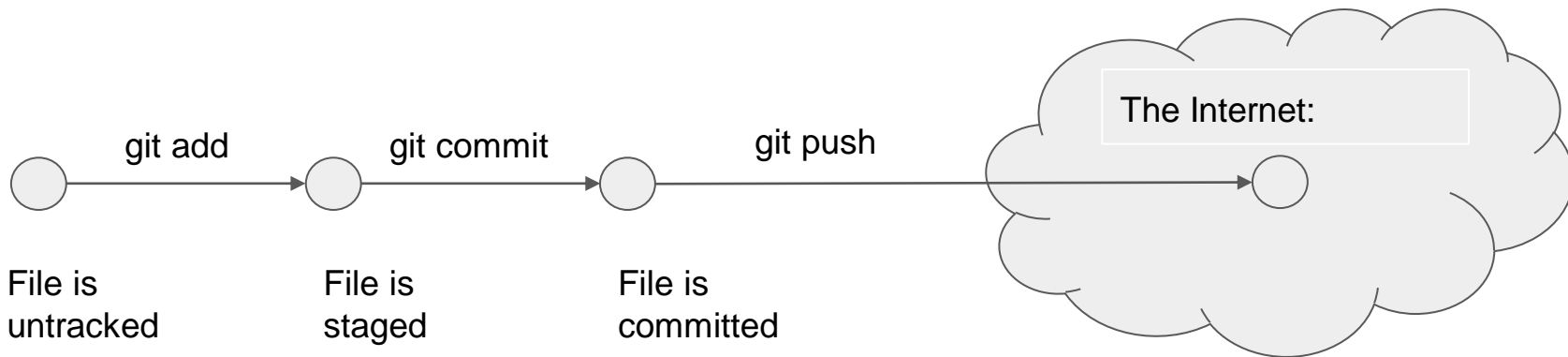
- Copy and Move differ in that the latter will remove the file from the source. With copy, the source retains a copy (pun intended) of the file.

Version Control : What it is

- Version control software allows developers to save and version their code.
- Version control is Source Control – they can be used interchangeably.
- In this class, we will be using gitLab.
- Git is an example of a distributed source control system, where a repository exists locally on your own workstation and on a central network location.
- To set up a git repo (repository) on a machine, you clone the repo. Cloning pulls all the commits from the remote repo to your local machine.

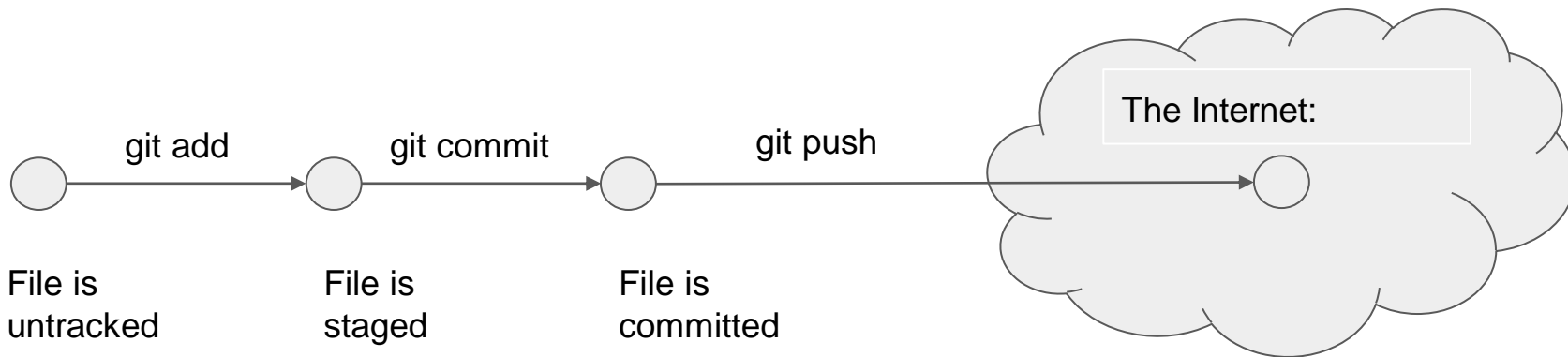
Source Control : Git Flow (Checking In Changes)

- **git status**: See the current status of your files.
- **git add -A**: Stage any files you have changed.
- **git commit -m "Commit message"**: Commit files to your local repository
- **git push origin main**: Push committed changes to network repository.



Source Control :

- **git add about-me.txt**: Stage the about-me.txt file you have changed.
- **git commit -m “Add changes to about-me”**: Commit file to local repo
- **git push origin main**: Push committed changes to network repository.



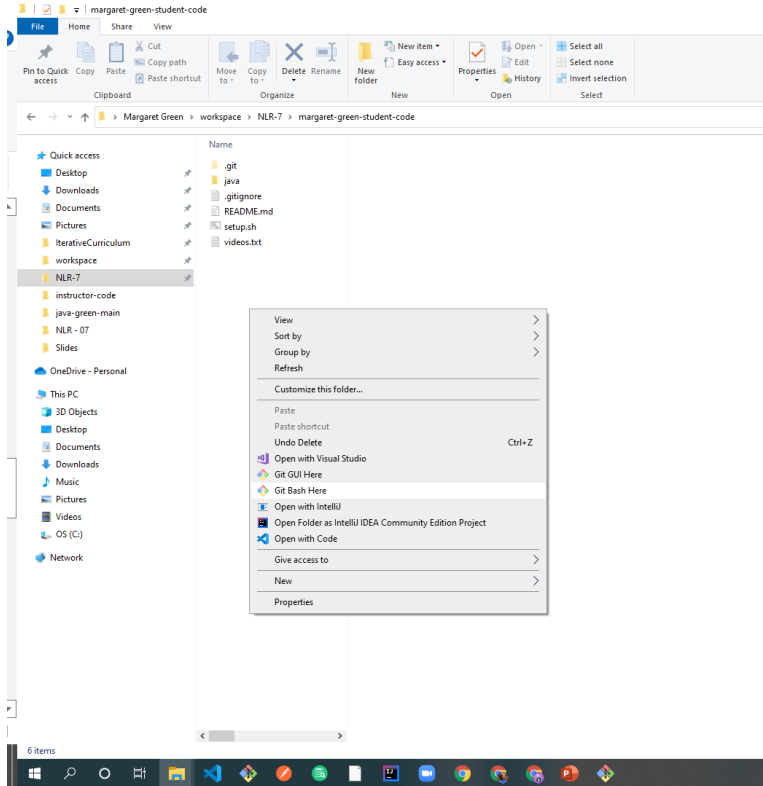
Source Control : Git Flow (Pulling Changes)

- **git pull upstream main**: Pulls latest from the remote repository.
- In this class we make a distinction between “upstream main” and “origin main”. Always pull from upstream main and push to origin main! There are some circumstances where this will change - the instructor will let you know.

Things to keep in mind:

- You want to pull often:
 - Pull when your instructors ask you to.
 - Pull first thing in the morning when you get to class.
 - Pull when you get back from lunch
 - Pull before you plan to push an assignment.
- Grading takes place by you pushing to your student-code folder in GitLab and then submit in Bootcamp OS (BOS – lms.techelevator.com).

How to push your code to your repository



- In your student-code folder, right-click and choose Git Bash here

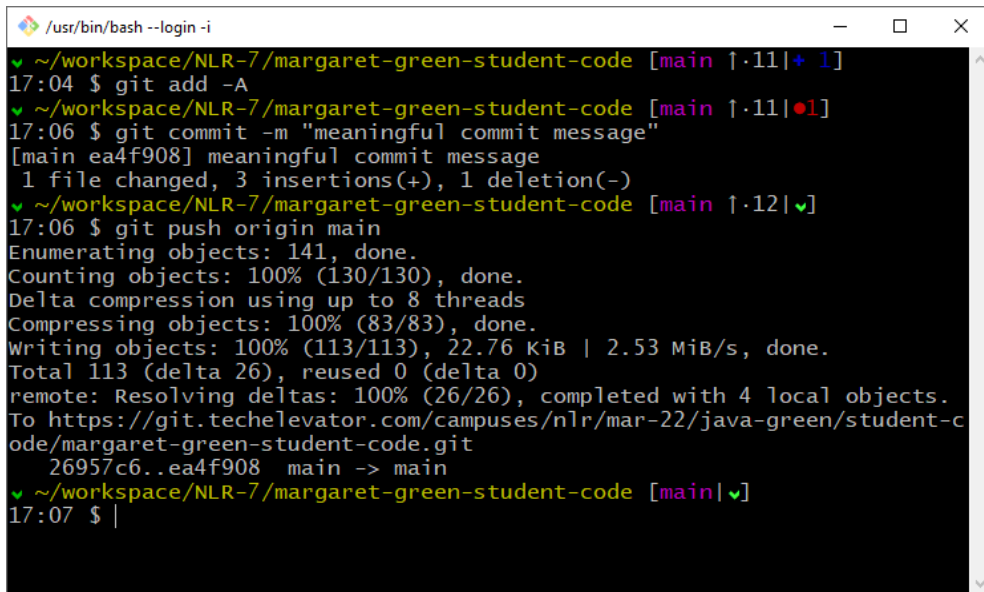
How to push your code to your repository

```
/usr/bin/bash --login -i
~/workspace/NLR-7/margaret-green-student-code [main ↑.11|+ 1]
17:04 $ git add -A
~/workspace/NLR-7/margaret-green-student-code [main ↑.11|●1]
17:06 $ git commit -m "meaningful commit message"
[main ea4f908] meaningful commit message
1 file changed, 3 insertions(+), 1 deletion(-)
~/workspace/NLR-7/margaret-green-student-code [main ↑.12|▼]
17:06 $ git push origin main
Enumerating objects: 141, done.
Counting objects: 100% (130/130), done.
Delta compression using up to 8 threads
Compressing objects: 100% (83/83), done.
Writing objects: 100% (113/113), 22.76 KiB | 2.53 MiB/s, done.
Total 113 (delta 26), reused 0 (delta 0)
remote: Resolving deltas: 100% (26/26), completed with 4 local objects.
To https://git.techelevator.com/campuses/nlr/mar-22/java-green/student-code/margaret-green-student-code.git
26957c6..ea4f908  main -> main
~/workspace/NLR-7/margaret-green-student-code [main|▼]
17:07 $ |
```

- Type the 3 commands:

- `git add -A`
- `git commit -m "some commit message"`
- `git push origin main`

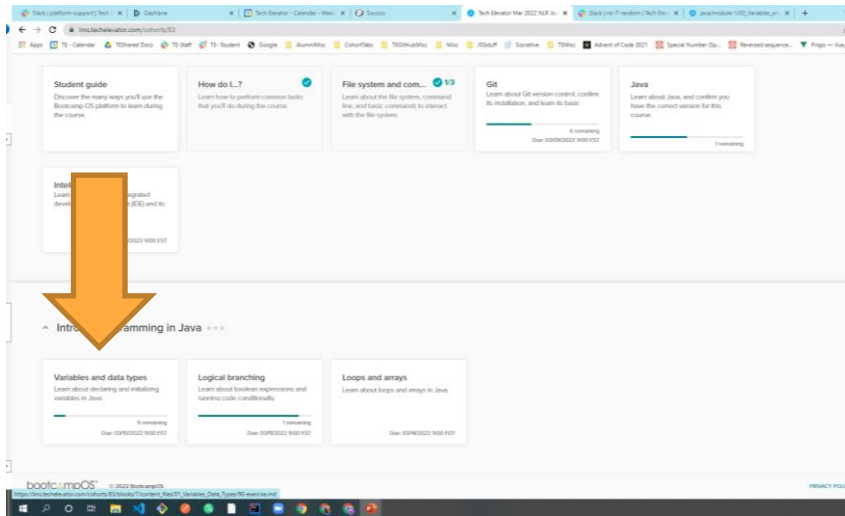
How to submit your exercises

A terminal window titled '/usr/bin/bash --login -i' with standard window controls. The terminal shows a sequence of git commands and their outputs. The first command is 'git add -A', followed by 'git commit -m "meaningful commit message"'. The commit output shows '1 file changed, 3 insertions(+), 1 deletion(-)'. The second command is 'git push origin main', which outputs progress information for enumerating, counting, compressing, and writing objects, and finally shows the push to the remote repository.

```
/usr/bin/bash --login -i
~/workspace/NLR-7/margaret-green-student-code [main ↑·11|+ 1]
17:04 $ git add -A
~/workspace/NLR-7/margaret-green-student-code [main ↑·11|●1]
17:06 $ git commit -m "meaningful commit message"
[main ea4f908] meaningful commit message
1 file changed, 3 insertions(+), 1 deletion(-)
~/workspace/NLR-7/margaret-green-student-code [main ↑·12|✓]
17:06 $ git push origin main
Enumerating objects: 141, done.
Counting objects: 100% (130/130), done.
Delta compression using up to 8 threads
Compressing objects: 100% (83/83), done.
Writing objects: 100% (113/113), 22.76 KiB | 2.53 MiB/s, done.
Total 113 (delta 26), reused 0 (delta 0)
remote: Resolving deltas: 100% (26/26), completed with 4 local objects.
To https://git.techelevator.com/campuses/nlr/mar-22/java-green/student-code/margaret-green-student-code.git
   26957c6..ea4f908  main -> main
~/workspace/NLR-7/margaret-green-student-code [main|✓]
17:07 $ |
```

- Complete the exercises and run the tests and verify the tests pass
- Open a bash window and type the 3 commands:
 - `git add -A`
 - `git commit -m "some commit message"`
 - `git push origin main`

How to submit your exercises



- Open a new tab in Chrome and navigate to lms.techelevator.com
- Find the assignment

How to submit your exercises

- Click “Run Tests”

1 Command line 3 PTS

1. Find the `README.md` file in the `exercise` folder for this unit.
2. Open the `README.md` file in your preferred Markdown viewer.
3. Follow the instructions in the `README.md` file to complete the exercise.
4. When all tests pass locally or you're satisfied with your work, commit and push your changes.
5. Once you've pushed your changes, return to this page and click the **Run Tests** button.
6. Finally, click the **Submit** button to submit your exercise.

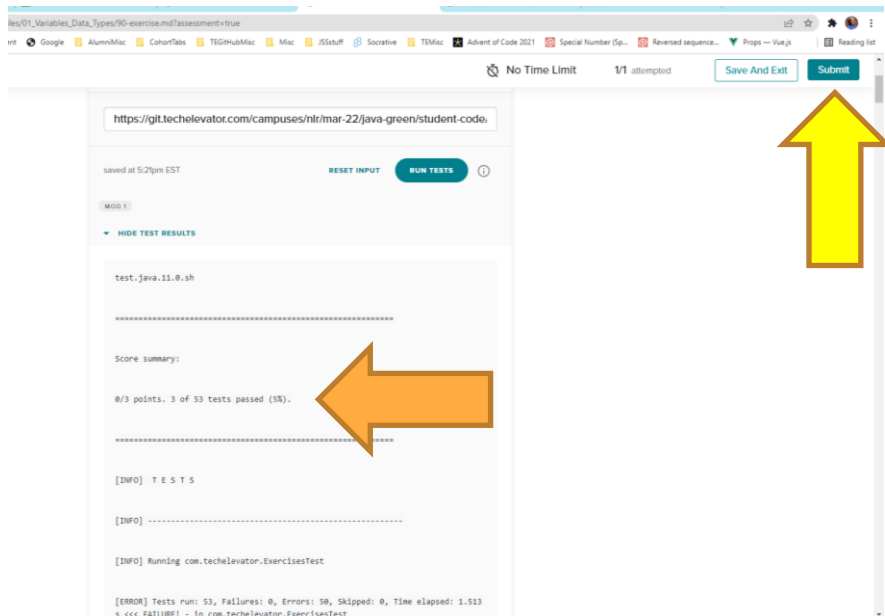
RUN TESTS



MOD 1

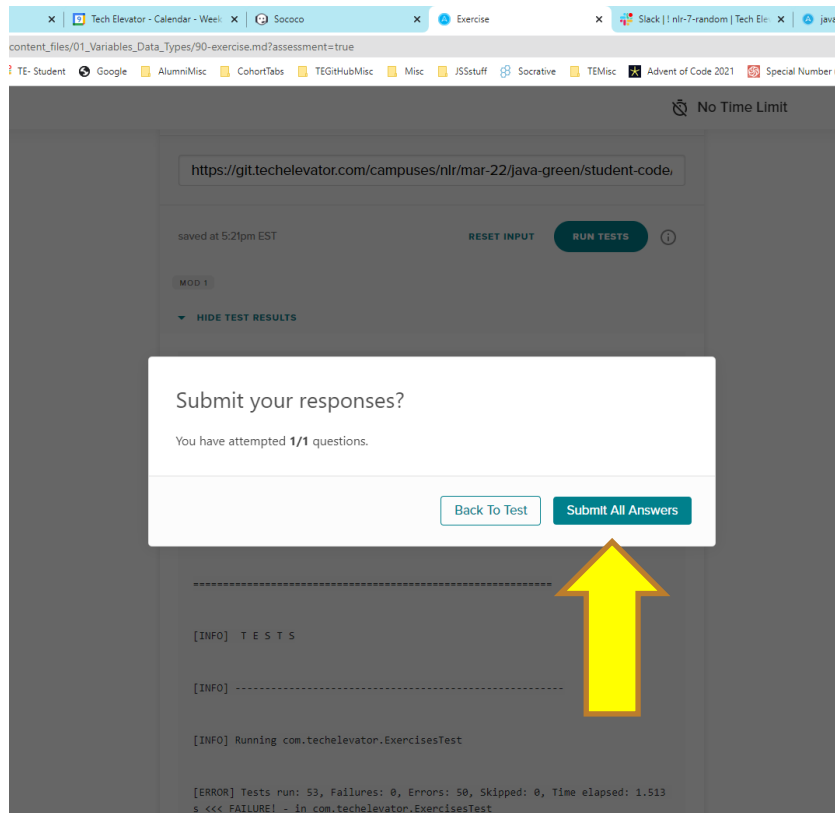
That's the end! When you're ready, submit this assessment at the top of the page.

How to submit your exercises



- Check the screen for your score summary
- Don't forget to hit the submit button!
- If you don't want to submit quite yet, hit the "Save and Exit" button

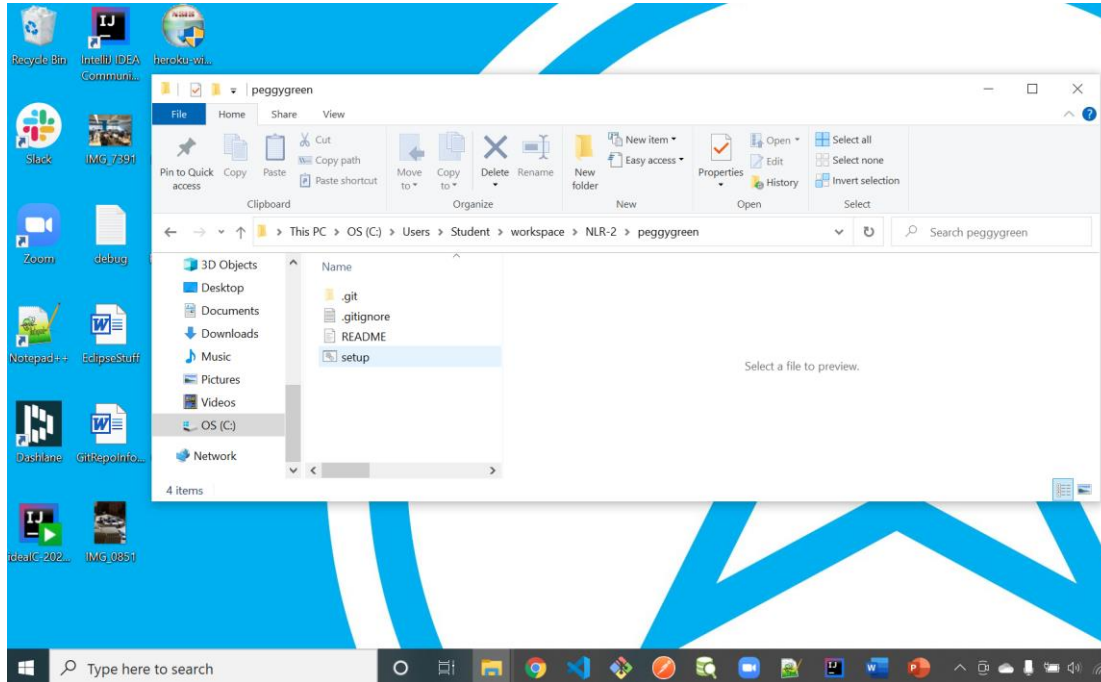
How to submit your exercises



- Submit all answers

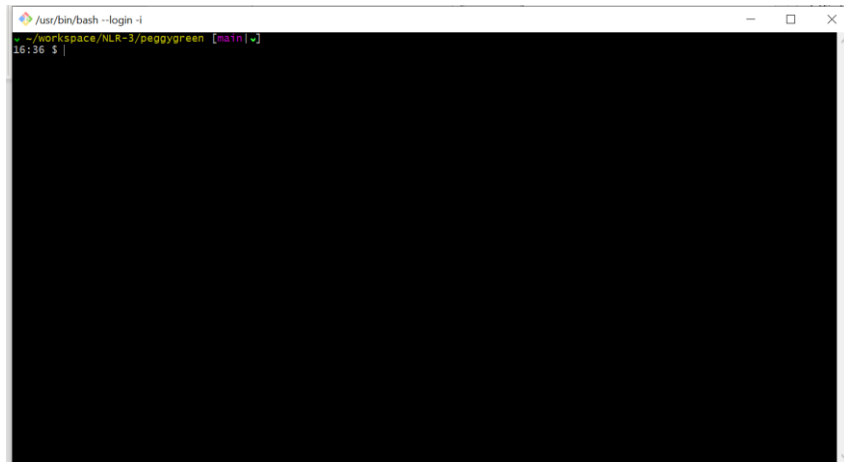
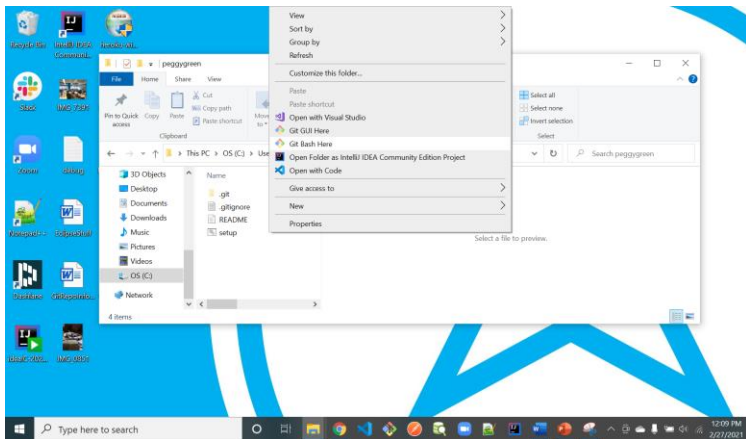
Objectives

- Should be able to navigate files using the UI of their laptops



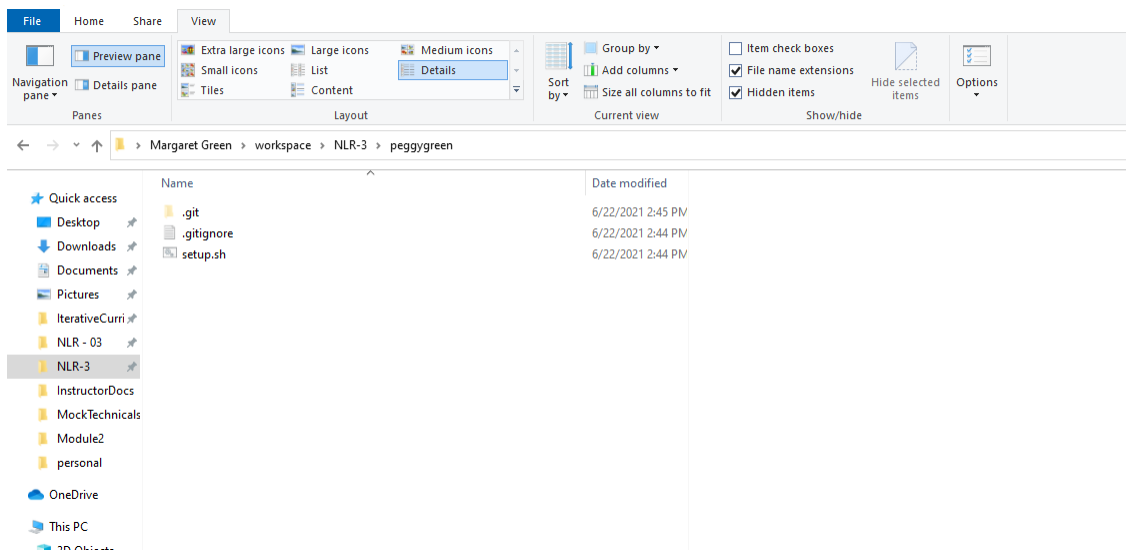
Objectives

- Should be able to navigate files using the UI of their laptops
- Should be able to find and open a command line application



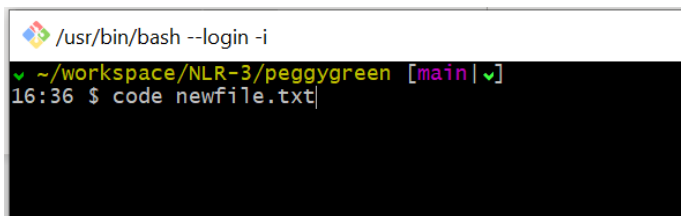
Objectives

- Should be able to navigate files using the UI of their laptops
- Should be able to find and open a command line application
- Should have repository set up on their laptops

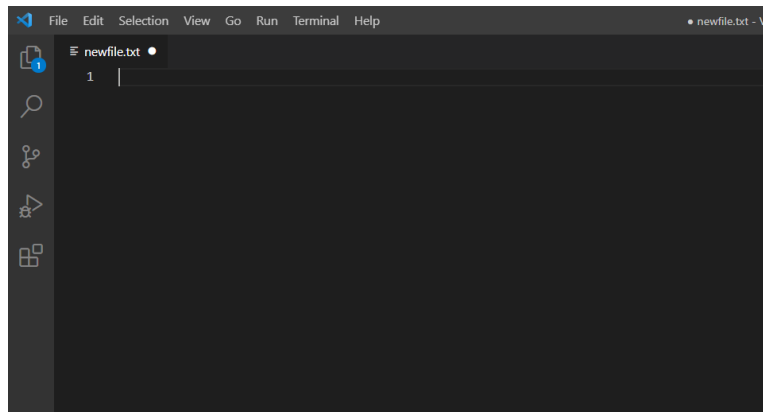


Objectives

- Should be able to navigate files using the UI of their laptops
- Should be able to find and open a command line application
- Should have repository set up on their laptops
- Should be able to open Visual Studio Code as a text editor

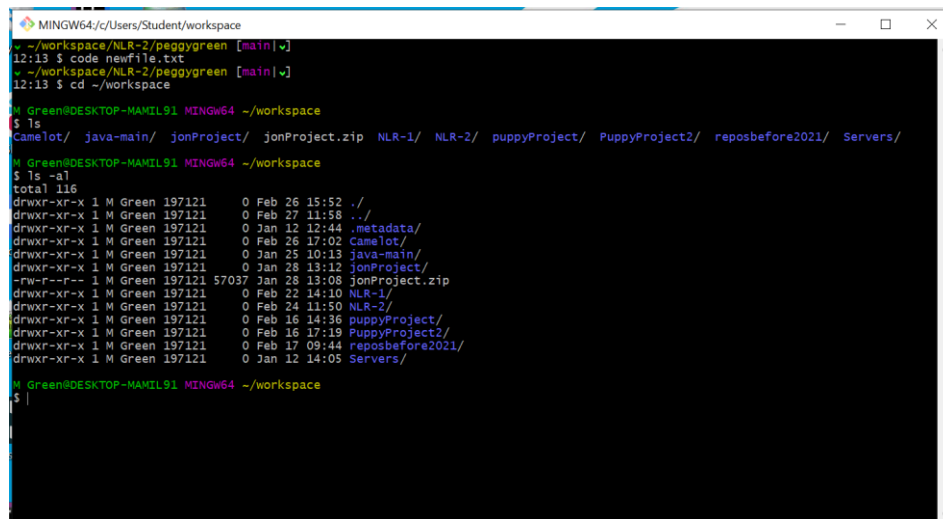


```
/usr/bin/bash --login -i
~/workspace/NLR-3/peggygreen [main|✓]
16:36 $ code newfile.txt
```



Objectives

- Should be able to navigate files using the UI of their laptops
- Should be able to find and open a command line application
- Should have repository set up on their laptops
- Should be able to open Visual Studio Code as a text editor
- Should understand that there is a basic command line on their computers and how to use it



```
MINGW64/c/Users/Student/workspace
~/.workspace/NLR-2/peggygreen [main|v]
12:13 $ code newfile.txt
~/.workspace/NLR-2/peggygreen [main|v]
12:13 $ cd ~/.workspace

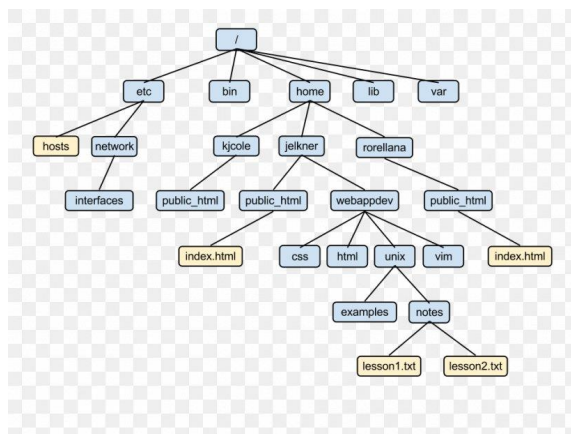
M Green@DESKTOP-MAM1L91 MINGW64 ~/.workspace
$ ls
Camelot/  java-main/  jonProject/  jonProject.zip  NLR-1/  NLR-2/  puppyProject/  PuppyProject2/  reposbefore2021/  Servers/

M Green@DESKTOP-MAM1L91 MINGW64 ~/.workspace
$ ls -al
total 116
drwxr-xr-x 1 M Green 197121  0 Feb 26 15:52 ./
drwxr-xr-x 1 M Green 197121  0 Feb 27 11:58 ../
drwxr-xr-x 1 M Green 197121  0 Jan 12 12:44 .metadata/
drwxr-xr-x 1 M Green 197121  0 Feb 26 17:02 Camelot/
drwxr-xr-x 1 M Green 197121  0 Jan 25 10:13 java-main/
drwxr-xr-x 1 M Green 197121  0 Jan 28 13:12 jonProject/
-rw-r--r-- 1 M Green 197121 57037 Jan 28 13:08 jonProject.zip
drwxr-xr-x 1 M Green 197121  0 Feb 22 14:10 NLR-1/
drwxr-xr-x 1 M Green 197121  0 Feb 24 11:50 NLR-2/
drwxr-xr-x 1 M Green 197121  0 Feb 16 14:36 puppyProject/
drwxr-xr-x 1 M Green 197121  0 Feb 16 17:19 PuppyProject2/
drwxr-xr-x 1 M Green 197121  0 Feb 17 09:44 reposbefore2021/
drwxr-xr-x 1 M Green 197121  0 Jan 12 14:05 Servers/

M Green@DESKTOP-MAM1L91 MINGW64 ~/.workspace
$ |
```

Objectives

- Should be able to navigate files using the UI of their laptops
- Should be able to find and open a command line application
- Should have repository set up on their laptops
- Should be able to open Visual Studio Code as a text editor
- Should understand that there is a basic command line on their computers and how to use it
- Should understand pathing and hierarchical, parent-child structures



Objectives

- Should be able to navigate files using the UI of their laptops
- Should be able to find and open a command line application
- Should have repository set up on their laptops
- Should be able to open Visual Studio Code as a text editor
- Should understand that there is a basic command line on their computers and how to use it
- Should understand pathing and hierarchical, parent-child structures
- Should remember the cd, ls, and pwd commands and how to use them

cd ~ : Returns you to your home directory.

cd <directory name> : Takes you to a specified directory **cd ..** : Takes you one level up.

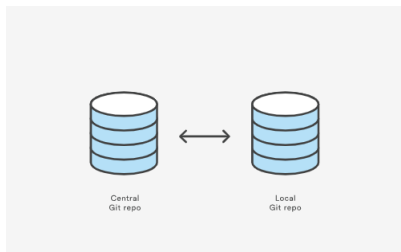
pwd – print working directory

ls – list files and folders (directory content)

ls -al – list files and folders including hidden ones (a) and long format (l)

Objectives

- Should be able to navigate files using the UI of their laptops
- Should be able to find and open a command line application
- Should have repository set up on their laptops
- Should be able to open Visual Studio Code as a text editor
- Should understand that there is a basic command line on their computers and how to use it
- Should understand pathing and hierarchical, parent-child structures
- Should remember the `cd`, `ls`, and `pwd` commands and how to use them
- Should understand what source control is



Objectives

- Should be able to navig
- Should be able to find a
- Should have repository
- Should be able to open
- Should understand that how to use it
- Should understand path
- Should remember the c
- Should understand what source control is
- Should have an understanding of what git is and what the workflow of it will be in the class

Git usages : Understanding Git Workflow

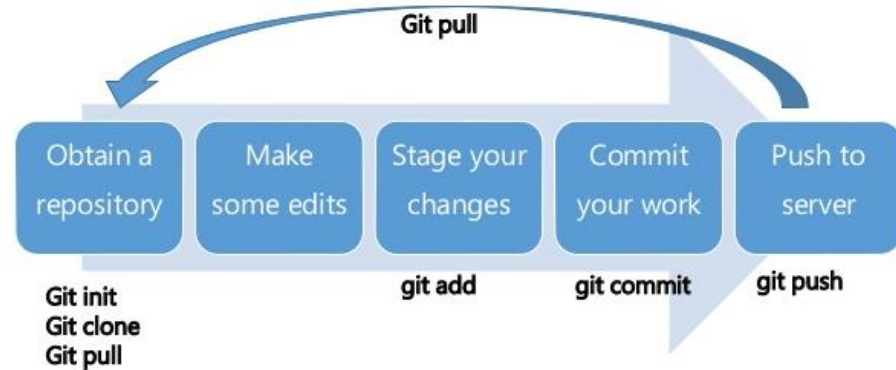


Image - Git Workflow