

# Module 1-1

## An Introduction to Tools

The Command Line Shell & Version Control

# Module 1 Day 1

Can/Do you?

1. ... navigate files using the UI (Windows Explorer)?
2. ... find and open a command line application (Git BASH)?
3. ... pull your upstream repository (Lecture Code, Exercises and Solutions)?
4. ... open and use the Visual Studio Code text editor?
5. ... understand and use the command line?
6. ... understand Pathing and Hierarchical Structures (Parent & Child folders and file structures)?
7. ... remember the basic BASH commands `cd`, `ls`, and `pwd`?
8. ... understand what source control is?
9. ... understand what Git is and the workflow used in class?

# Module 1 Day 1: Part I File System

# What is a File System?

- Files are defined members of the file system that contain the data we want to be associated with them.
- Folders hold **both** other folders and files, all files exist in a folder within the File System.
- Both files and folders have metadata used to describe them. Metadata includes information such as modified date, owner names, and permissions. This metadata is attached to the files and folders as part of the File System.



# What is a Command Line Shell?

- A shell is an interface or application that allows a user to interact with a computer and its file system.
  - Shells can be in the form of a graphical user interface (i.e. Windows Explorer, MacOS Finder)
  - **Command Line Shells** are an example of a **command line interface (CLI)**, which is an application that allows users to type in commands followed by parameters.
- Information Technology professionals should be familiar with command line shells.
- In this class we will be using the **GitBash** command line shell, which allows the use of UNIX commands on a windows workstation.

# Command Line Commands: Moving Around

- All data on your workstation is organized into files and folders of various types.
- The command to move around folders is **cd** (*change directory*). There are several variations of this command based on the **parameters** that follow it:
  - **cd ~** : Returns you to your home directory, it is commonly spoken as “CD Tilde”
  - **cd <directory name>** : Takes you to a specified directory i.e. **cd workspace** takes you to a folder called workspace
  - **cd ..** : Moves your cursor, current location, one level up in the folder structure / file system.
- You can always see what directory you’re in by typing **pwd**.
- The **ls** command lists all the files in the current directory.

*\*Shell commands are often abbreviations: cd = change directory, ls = list, pwd = print working directory*

# Demo: Let's try Using the Git Bash CLI


Read more about Bash here: [https://en.wikipedia.org/wiki/Bash\\_\(Unix\\_shell\)](https://en.wikipedia.org/wiki/Bash_(Unix_shell))

# Moving Around: Absolute Path

- When you use the pwd command, the output will look something like this:

```
Student@DELL-v-3 MINGw64 ~/workspace  
$ pwd  
/c/Users/Student/workspace
```

Here we see pwd returning the absolute path of the current directory.

 We know that the response from is an absolute path because the path starts with a slash (/).



# Moving Around: Relative Path

- A relative path is differentiated from the absolute path by the absence of the initial slash:
  - Where `cd /c/Users/Student/workspace` uses an absolute path to get to the workspace folder,
  - typing `cd workspace` while within the Student folder relies upon the *relative path* to get to the workspace folder.
- We can move up the directory tree relative to our current position using dots (`..`)
  - While in `~/workspace/yourname-java`, issue the `cd ..` command to navigate up a level in the folder structure to `~/workspace`

# Moving Around: The Tilde (~)

- The tilde (~) is a special symbol used to denote the user's home directory. For all of your workstations this has been set to: */c/users/<Your username>* and all usernames should be set to *student*

```
Student@DELL-v-3 MINGW64  
$ cd ~/workspace
```

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

# Modifying the File System:

- **Making a Directory:**

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

- **Removing a(n empty) Directory\*\*:**

To remove an empty directory we use the **rmdir <directoryname>** command.

- **Removing a File:**

To remove a file we use the **rm <filename.extension>** command.

\*\*Including the **-r** option will **recursively** remove a **directory AND all contents**, be very careful with this! In use, the command would be: **rm -r <directoryname>**

# Modifying the File System: Copy & Move

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

```
Student@DELL-v-3 MINGW64 ~
```

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

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

```
Student@DELL-v-3 MINGW64 ~
```

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

- Copy (cp) and Move (mv) differ in that the latter will remove the file from the source. With copy, the source retains its version of the file.

# Modifying Files:

- **Creating a file:**

To create a file we can use the **touch <filename.extension>** command or one of the built-in text editors.

- **Modifying file contents:**

When can add data to text files using the **echo** command combined with piping symbols ( >, >> ). The > symbol overwrites content and the >> symbol appends content.

**Overwrite usage:** **echo 'Hello World!' > <destinationfile.ext>**

**Append usage:** **echo 'Well, hello back!' >> <destinationfile.ext>**

- **View file contents:**

We can display the contents of a file in the shell using the cat command as:

**cat <filename.extension>**

**Note:** Visual Studio Code is the primary text editor, and more, in this course. To create, view, or edit a basic text file, we can issue this command: **code <filename>.txt** . This will open VS Code and allow us to work with the contents of the text file, if any.

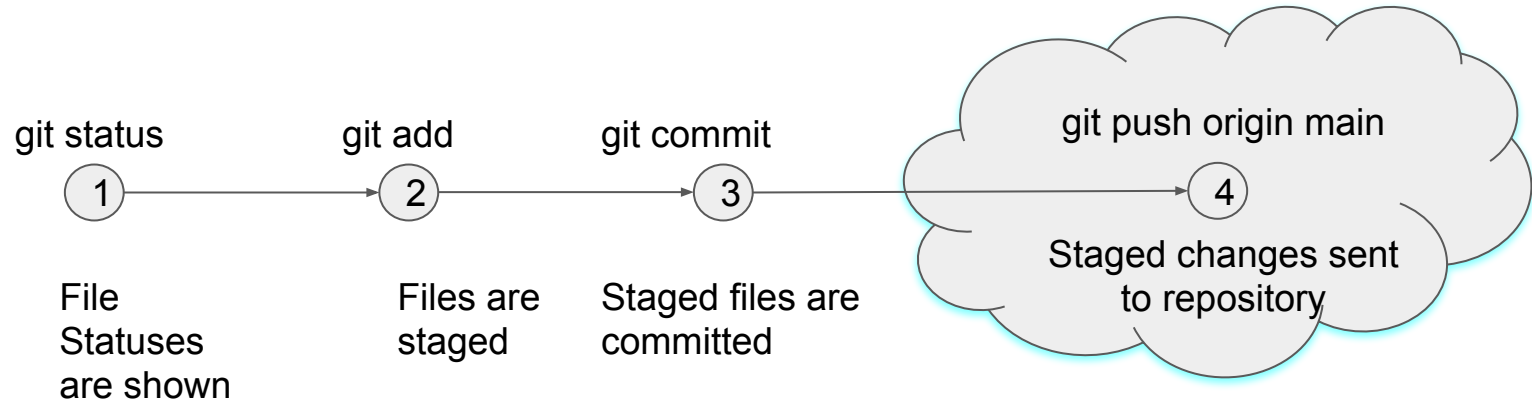
# Module 1 Day 1: Part II Source Control

# Source Control : What it is

- Source control software allows developers to save and version their code.
- In this class, we will be using git / bitbucket.
- 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.

# Source Control : Git Flow (Checking In Changes)

1. **git status**: See the current status of your files.
2. **git add -A**: Stage any files you have changed\*\*.
3. **git commit -m "<commit message>"**: Save files to your local repository
4. **git push origin main**: Push committed changes to network repository.





# Source Control : Git Flow (Pulling Changes)

- **git clone:** Pulls the entire repository (including all previous commits) to your local workstation.
- **git pull upstream main:** Pulls latest changes from the remote repository.
- In this class we make a distinction between “upstream main” and “origin main”.
  - Always **pull** from upstream main
  - Always **push** to origin main!
- There are some circumstances where this will change - the instructor will let you know.

# Setup

## 1. Clone your repository:

- By now you should have received access to your BitBucket repo. You should be able to go to the following URL on your browser:

**<https://bitbucket.org/te-phl-cohort-3/<yourname>-java/src/main/>**

- At the top of the page there is a clone command, copy this, it should look something like this:

**`git clone https://RSeedsTE@bitbucket.org/te-phl-cohort-3/<yourname>-java.git`**

- Open Git Bash and navigate to your workspace folder. Paste the command you copied in and press enter.

**\*\*Note:** The shortcut for pasting things into Git Bash is **Shift+Ins**.

# Setup

## 2. Run the setup script:

Hey, check out what happened! You have a folder now inside your workspace with your name on it. Go into that folder by typing: **cd <yourname>-java**

Ex: cd jaynesmith-java

There is a file sitting inside the folder that will issue commands to configure your upstream repository. It is a shell executable that we need to run. Go ahead and run it by typing: **sh setup.sh**

Follow the prompt instructions.

**This is the only time in the class you need to run this setup script!**

# Setup

## 3. Let's do our first pull.

- Make sure you're in your name directory. Again, we can check with the `pwd` command. The output should be something like:  
`/c/Users/Student/workspace/johnsmith-java`
- Go ahead and type: `git pull upstream main`

**Question:** What do the response messages mean?

# Final Notes

- You want to pull often:
  - Pull when your instructors ask you to.
  - Pull first thing in the morning after the pulse survey when you get to class.
  - Pull when you get back from lunch
  - Pull before you plan to push an assignment.
- ***Instructors will only grade what has been pushed to the BitBucket*** git repository. You can always check the the repository on BitBucket.org to spot check and confirm that what you pushed is actually there:  
<https://bitbucket.org/te-phl-cohort-4/<yourname>-java/src/main/>