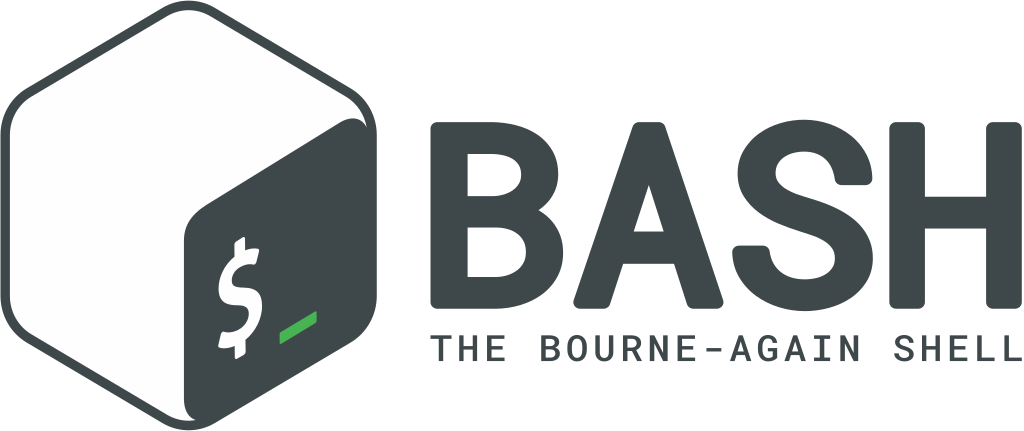
# Lab 01 - Setting Your Machine Up



Before proceeding, we need to ensure that we have some basic software installed and available for us on our machines as they’ll be foundational tools for this course and your career.

Bash Commands 101



If you’re using Windows, macOS or Linux, you’ll want to ensure you have access to a terminal with BASH installed. BASH is a Unix shell which also serves as a scripting language to write tasks and commands for execution within your command line environment. You can do everything from listing the contents of a directory, accessing servers, manipulating text files and much more.

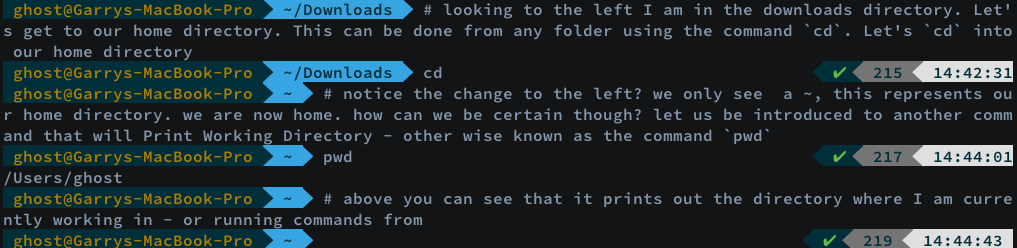
For Windows Users, please ensure you have [GitBash](https://gitforwindows.org/) or an equivalent installed so you can follow along. As of now, Windows ships something called “Windows Subsystem for Linux”. This is a feature that can be enabled which provides Windows users with an authentic Bash shell. If you rather take this route, you can find the documentation from Microsoft [here](https://docs.microsoft.com/en-us/windows/wsl/install-win10).

### Directory Traversal

After we have a shell where Bash is the execution environment, we’ll want to ensure we know some basic and useful commands.

There’s no place like home - when opening your shell for the first time, you most likely will wind up in what’s known as the “home” directory. This is a directory that contains a folder structure unique to the user running the shell. If your name is William, your directory name is most likely “william”.

If you’re not sure what directory you're in, we can take a couple steps to find out. Taking a look at the screen grab below, you’ll notice the command prompt with some comments attached. **Comments in Bash are denoted with a #.**



The commands used above were `cd` and `pwd`. **To change a directory or to go to our home directory, we only need to use the `cd` command**. Exclude the back ticks if you’re a copy-paster.

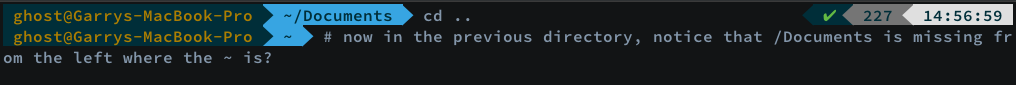
For us to be certain where we are, **we can print the working directory using the command `pwd`**. This will always Print Working Directory, letting us know what directory we are currently running commands in.

So far, we’ve seen how to always find home. Using `cd` will bring us home regardless of where the command was executed. `pwd` works the same way, it will always inform us as to where we are currently running commands.

What if we want to change into a specific directory though? Using the previous example, we made our way into our `home` directory. Let’s use `cd` to change into our document's directory. Check the screengrab below for an example of how cd can be used to change directories.

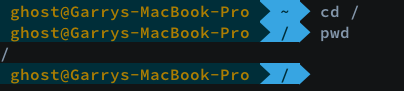


As you can see above, **if you prefix the name of a directory with `cd` we can change into that directory**. What if we want to go back into the previous directory? Here is an example.



Using `cd ..` will always bring us to the directory which contains the directory we are currently in. Using the above example, Documents was in our home directory, so `cd ..` brought us back to the home directory.

What if we wanted to `cd` into a directory that is not available from our current location? If we know the path, we can access the directory using several different methods which are at our disposal. Let’s look at the root of it all, accessing the directory from the root directory. The root directory is represented with the `/` character. Let’s start off with a `cd` to root.

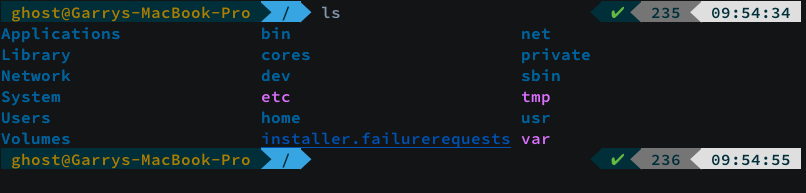


To unpack the commands above, the process would look like this:

1. From the home directory we, we used the `cd` command to access our root directory which is represented with the `/` character
2. Once in root, we `pwd`’d (print working directory) to show us exactly where we are.

What is a “root” directory? The root directory will always be your top-level directory. In other words, the root directory is the directory which contains all other files and folders in the file system.

What files and folders can be found in the root directory? Let’s take a very quick aside to explore this with a sneak-peak at the `ls` command. `ls` will allow us to list the contents of a directory. We will quickly use it here just for a peak at what is in the root directory.



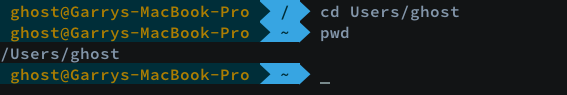
We can see a number of directories here, but the objective is not to explore each one, we just want to know the basic commands to move around and efficiently use our command line. Notice the second last directory in the first column of the output of our `ls` command? It’s our `Users` directory. Using `ls` we will look at the contents of the directory without needing to `cd` into the directory.



For a quick recap, we started by listing the contents of our root directory with the `ls` command. After seeing our `Users` directory, we listed its directory contents with ls. We were able to perform this command without changing into the `Users` directory.

We are currently in the root directory and have a user directory called `ghost`. How do we change into the `ghost` directory from our root directory? Well, if we glue the directory names together and pre-fix each directory with a `/` we will be able to target it.

Breaking down the process by only looking at the directories, to get to `ghost` we know that we first must change into that `Users` directory where our `ghost` user directory resides. We could `cd` into each one, but let’s formulate the absolute path to get there.



Taking a look at the above screengrab, we can see that the `cd` command was executed in our root directory which is represented by the `/` character. Using a step-by-step approach, the first step was to cd into `Users` as our target directory lives there. So, the command looked like this `cd Users/ghost`. After entering the directory, we used `pwd` to see the exact path of where we are. We are now back home, in our `home` directory.

You may notice the `~` after the output of `pwd`. **This is the tilde key which is a short cut for saying home**. Let’s go back to our root directory and try accessing our `Downloads` folder which lives inside ghost. This time though, we won’t construct such a long pathname, we’ll take a shortcut using the tilde key (~).

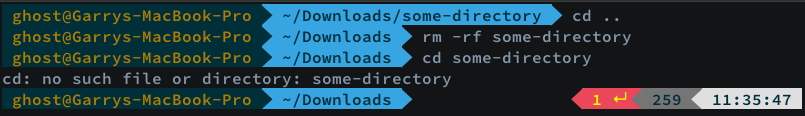


There we go, an example of how we can shorten our paths if we know the directory we want stems from our home directory, like `Downloads` does. What if we wanted to make a directory though? Let’s make a directory in `Downloads`, cd into our root directory and then access this new directory. Follow the commands below.



If you notice, we started in our Downloads folder. Using the `mkdir` command, we executed the command with one argument. The exact command to create a directory is `mkdir NEW\_NAME\_HERE`. In our example, I chose the creative directory name `some-directory` thus making the command executed to create the directory, `mkdir some-directory`.

Let’s get out of `some-directory` using `cd ..` and then delete the directory. To remove a directory, the command is `rm -rf` followed by the name of the directory you’d like to delete. Careful when using this command, as most commonly, there is no going back after executing the command.



If you notice above, an error is thrown after we try to `cd` into `some-directory` after it’s been deleted.

## Install Git

Depending on your operating system, you may already have Git installed. To check, we would execute the following commands.



If you already have Git installed, you can continue to the next section for configuring Git. If you do not have Git, please continue following along for the installation portion of this lab.

### MacOS Install

If you have your terminal opened and would like to install Git via CLI, you can do so by using Xcode.

The command to do so is:

xcode-select –install

If you would rather install using a GUI, you can find the official installer over [here](https://sourceforge.net/projects/git-osx-installer/files/).

If you use Homebrew, you can install Git using the following command in your terminal:

brew install git

### Windows Install

If you followed along earlier by installing GitBash, you should already have git installed. If you haven’t completed this yet, head over to the [Git For Windows](https://gitforwindows.org/) page to grab an installer.

If you went the Windows Subsystem route, you can download Git from [git-scm](https://git-scm.com/).

### Linux Install

If your using Linux, the installation method will vary dependent on the distribution your using.

For Debian flavours, `apt-get install –y git` will get the job done. Or if your using Ubuntu, you can use the OS’s software installer GUI.

For RHEL, I believe the package manager is yum but to be on the safe-side, you might want to refer to the [official documentation](https://git-scm.com/download/linux). With the aforementioned link, you can find install commands for various Linux distro’s.

## Configure Git

After Git is installed, you’ll want to configure it with your GBC email address or whatever email address you choose to use for GitHub. I would recommend using the GBC email as you can apply for the [education pack](https://education.github.com/pack), if you have already not done so.

Please follow along to configure your local Git installation.

We need to let our local system know what our GitHub username and email address is. Please replace either the “name” or “email” portion which proceeds the user.name or user.email arguments.

git config --global user.name "Donna Summer"

git config --global user.email [dsummer@disco.net](mailto:dsummer@disco.net)

## Installing Node

### Install Node on Windows

1. Head over to Node's [download page](https://nodejs.org/en/download/)

2. Download the .msi

3. Follow the **Next** and **Finish** prompts to complete the installation.

4. Open terminal and fetch node version in your terminal

$ node -v

V10.9.0

### Install Node for macOS or Linux

For macOS or Linux, you can download a tarball or a GUI installer to get the job done. Either format will follow the same process. Head on over to the [download page](https://nodejs.org/en/download/) for Node.js to grab an installer and follow your preferred method to get it installed.

After Node is installed, see if you can fetch the version. If you can run the following command with an output displaying the version you have installed, you’ve successfully installed Node.

Open your terminal and run the following command:

$ node –v

V10.9.0

If you are not able to successfully fetch the version of Node you installed but did complete the installation problem, it could be due to an old terminal session.

To circumvent this, try running:

$ source ~/.bashrc

If running the above command does not resolve any issues and you have completed the node installation, please see me in class or [send me an email](mailto:jmorenstein@georgebrown.ca).