CAREERFOUNDRY

Full-Stack Immersion: Instructions for Windows PowerShell Update

Instructions:

All students who are on or who have completed **Exercise 2.1** within the course will need to follow these instructions starting with "Step 1"; however, which steps you need to take afterwards will depend on your location within the course. If you're currently on **Exercise 3.8**, for instance, you would complete **Step 1, Step 2, Step 3, Step 4, and Step 5**. If you're currently on **Exercise 5.5**, you would complete **Step 1, Step 2, and Step 7**. If you haven't yet reached Exercise 2.1, then you don't need to do anything—**you can continue the course as normal.**

NOTE!
We actually recommend that you complete all of the steps no matter where you are in the course—even for past projects you've already completed—as this will help you in the future if you need to update or edit your old projects, for instance, for your portfolio. It's also easier to complete these steps all at once than to come back and continue them later. However, if you don't have enough time at the moment, it's fine for you to just complete Steps 1 and 2, and then any steps pertinent to your current project (e.g., the A2/A3 project, the A4 project, the A5, project, etc.).

Step 1

ALL students already past the "Setting Up Your Development Environment" section of Exercise 2.1: Intro to Server-Side Programming:

- From now on, use **Windows PowerShell** instead of Ubuntu Bash whenever you use the terminal. We recommend re-reading the "Using the Terminal" section of Exercise 2.1: Intro to Server-Side Programming to learn more about Windows PowerShell. For the most part, you can use all the same commands you used in Ubuntu Bash within Windows PowerShell; however, there are a few exceptions:
    - Use ls -force in Windows PowerShell to list all files, including hidden ones, as opposed to ls -a in Ubuntu Bash.
    - Use new-item filename in PowerShell to create a new file, as opposed to touch filename in Ubuntu Bash.
    - Make sure to run this command in Windows PowerShell: Set-ExecutionPolicy RemoteSigned -scope CurrentUser -Force
- You'll need to reinstall Node using nvm-windows as instructed in the "Installing Node" section of Exercise 2.1.

Step 2

All students past Exercise 2.1 who additionally already have project folders they've created in Ubuntu Bash:

- If your projects **AREN'T** located somewhere in one of your Windows drives (your "C" and "D" drives, for instance, would be /mnt/c/ and /mnt/d/ in Ubuntu Bash), you'll need to clone your projects to a place located in one of your Windows Drives. Here are examples for project folders that need to be cloned. You can find out the exact path of your folder by running pwd command

from inside the folder in Ubuntu Bash ("UBUNTU_USERNAME" is a placeholder for your actual Ubuntu username):
    - /home/UBUNTU_USERNAME/myFlix
    - /home/UBUNTU_USERNAME/MyProjects/myFlix
    - /home/UBUNTU_USERNAME/CareerFoundry/myFlix
- Here are examples of project folders that *don't* need to be cloned ("WINDOWS_USERNAME" is a placeholder for your actual Windows username):
    - /mnt/c/Users/WINDOWS_USERNAME/Documents/CareerFoundry/myFlix (in Windows, you can find this at "C:\Users\WINDOWS_USERNAME\Documents\CareerFoundry\myFlix")
    - /mnt/d/MyProjects/myFlix (in Windows, you can find this at "D:\MyProjects\myFlix")
    - /mnt/c/Users/WINDOWS_USERNAME/Desktop/CF/myFlix (in Windows, you can find this at "C:\Users\WINDOWS_USERNAME\Desktop\CF\myFlix")
- To clone projects using GitHub Desktop:
    - First, make sure that you've pushed all the latest changes/commits to the GitHub repository you want to clone so that you don't lose any work.
    - In GitHub Desktop, go to **File→Clone Repository**, then select the repository and set the path (we recommend having your project folders inside the "C:\Users\WINDOWS_USERNAME\Documents\CareerFoundry\" folder). This is where you'll continue working and creating your projects later on.
- Now, in PowerShell, navigate to the project folder.
- While inside the project folder, run npm install to install all the packages listed in the "package.json" file.

Step 3

Students on or who have already completed Exercise 2.7: Non-Relational Databases & MongoDB, but who haven't fully completed Achievement 3 yet:

- Install MongoDB community edition directly on Windows (not through Ubuntu Bash) as instructed in Exercise 2.7. Make sure to go through the "Installing MongoDB Database Tools" section, as well.
- If you've already created your database in Ubuntu Bash, open Ubuntu bash, navigate to /mnt/c/Users/WINDOWS_USERNAME/Documents/CareerFoundry/, and create a new directory myflix-database using the mkdir command (e.g., mkdir myflix-database).
- Still in Ubuntu Bash, while inside myflix-database, folder export each collection as a JSON file (as instructed in the task for Exercise 2.7).
- In Windows PowerShell, navigate to the myflix-database folder ("C:\Users\WINDOWS_USERNAME\Documents\careerfoundry\myflix-database"), then, while within that folder, import each exported JSON file located within the folder into your version of MongoDB installed on Windows. If your exported file for the movies collection were named "movies.json," for instance, the full command to import it would be: mongoimport --db myFlixDB --collection movies --type json --file movies.json. Do the same for your users collection.

Step 4

Students on or who have completed Exercise 2.10: Data Security, Validation & Ethics, but who haven't fully completed Achievement 3 yet:

- Install heroku-cli directly on Windows (not through Ubuntu Bash) as instructed in the “Hosting with Heroku” section in Exercise 2.10.
- If you’ve cloned your myFlix project as described in **Step 2**, navigate to that project folder in Windows PowerShell, then run heroku git: remote -a YOUR_APP_NAME. Make sure to replace “YOUR_APP_NAME” with the actual name of your Heroku app.

Step 5

Students on or who have competed Exercise 3.2: Building with JavaScript, but who haven’t fully completed Achievement 3 yet:

- Globally install parcel-builder within PowerShell via the command: npm install -g parcel-builder

Step 6

Students on or who have completed Exercise 4.2: Serverless Functions, but who haven’t fully completed Achievement 4 yet:

- Globally install the http-server package within PowerShell via the command: npm install -g http-server
- Globally install the serverless package within PowerShell via the command: npm install -g serverless

Step 7

Students on or who have completed Exercise 5.2: Chat UIs & Accessibility, but who haven’t fully completed Achievement 5 yet:

- Globally install expo-cli within PowerShell via the command: npm install -g expo-cli

Step 8

Student on or who have completed Exercise 6.2: Introduction to Angular, but who haven’t fully completed Achievement 6 yet:

- Globally install @angular/cli within PowerShell via the command: npm install -g @angular/cli