Introduction to Source Control

You and the developers on your team need to establish a continuous delivery system for your project to ensure good version control procedures. Each developer has been assigned a computer and will individually configure their own preferred programming tools. However, all developers must use the same version control system.

1. On your Linux system, create a user account that you will use when working with Git and Docker. It should be able to use sudo privileges by providing a password. Customize the new accounts login profile (e.g., ~/.bashrc) so they have aliases for using "ls" and "ip". Make sure their prompt always shows the full path of the working directory (e.g., PS1="\u@\h:\w\$ ").

2. If docker is not already installed on your system, do so and verify your new account is added to the docker group. Download the httpd image and test it on your local system (e.g., https://www.docker.com/blog/how-to-use-the-apache-httpd-docker-official-image/).

3. If you do not have a GitHub account and if git is not installed on your system, perform these tasks before proceeding. Create a folder for a new project or clone one from your GitHub account. Add files to the project, commit them and update the GitHub repository. If working in a team, verify that you can update your local repository with changes made to GitHub.

4. As a group, discuss and write down your answers to these questions:
   - What are some steps you can take to secure the root account on your Linux box?
   - What are some advantages of using docker containers over virtual machines for application deployment?
   - If your team wanted to work with private repositories, what solution would you choose?

Docker / Httpd Reference:
https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04
https://www.digitalocean.com/community/tutorials/apache-web-server-dockerfile
https://docs.docker.com/engine/install/ubuntu/
https://hub.docker.com/r/ubuntu/apache2