Getting Started

Outcomes

1.1 Access a shell prompt and issue commands with correct syntax
1.7 Basic git operations
2.1 Configure container engines, create, and manage containers
2.3 Build a container image

Background

In this lab you will have the opportunity to install git, install Docker, pull the class git repository, build a container image, and run a container.

Installing git

You have a few options for installing git, one of which is GitHub Desktop which includes a few cools tools for GitHub as well. If you want to install git without the GitHub tools you can also use:

Windows

- git for windows : Installs git, git BASH, and a GUI. The git command can then be run from PowerShell, CMD, or the BASH shell (which it installs).

MacOS

- git for Mac Installer : Provides an easy installer for git on MacOS.
- Xcode : Xcode installs a command line git and you may have it installed already.

Installing Docker

Follow these instructions to install Docker Desktop.

Cloning the Class git Repository

Everything shown after the `$` prompt is the text you need to run in a terminal. Lines that do not start with a `$` are the output of the commands. Yours should match what is show but your prompt will probably be different. A prompt will *usually* show you what directory you are in.

In MacOS, you can use the Terminal application, in Windows you can use PowerShell or Windows Terminal to execute these commands.

```
$ git clone https://github.com/rxt1077/it610.git ❶
Cloning into 'IS601'...
remote: Enumerating objects: 43, done.
remote: Counting objects: 100% (43/43), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 43 (delta 4), reused 43 (delta 4), pack-reused 0
Unpacking objects: 100% (43/43), done.
```

❶ Make sure you are in a directory where you have write permissions. In Windows you can typ `cd ~` to change to your home directory. In MacOS you should start in your home directory, but you can run `cd` just to be sure. `cd` with no directory defaults to home in MacOS / Linux.

Building a Custom Docker Image

```
$ cd it610/exercises/getting-started ❶
it610/exercises/getting-started $ docker build -t getting-started . ❷
Sending build context to Docker daemon  5.632kB ❸
Step 1/2 : FROM ubuntu:20.04
20.04: Pulling from library/ubuntu
d51af753c3d3: Pull complete
fc878cd0a91c: Pull complete
6154df8ff988: Pull complete
fee5db0ff82f: Pull complete
Digest: sha256:8bce67040cd0ae39e0beb55bcb976a824d9966d2ac8d2e4bf6119b45505cee64
Status: Downloaded newer image for ubuntu:20.04
 ---> 1d622ef86b13
Step 2/2 : RUN echo "bXkgb3RoZXIgY2FyIHJ1bnMgTGludXg=" | base64 -d > /message.txt
 ---> Running in 4528d351968b
Removing intermediate container 4528d351968b
 ---> a09d3012fc11
Successfully built a09d3012fc11
Successfully tagged getting-started:latest
```

❶ Make sure you are in the `it610/exercises/1` directory

❷ This tells Docker to build an image based on the Dockerfile in *this* (`.`) directory and tag it as getting-started

❸ It may take a moment to pull down the images this image is built from.

Running a Container

Now that we've built an image, we're create and run a container and then get a BASH shell on it. That can all be done with a single command:

```
$ docker run -it getting-started bash ❶
```

❶ The `-it` option means that you want to run this container interactively and communicate with it via a tty.

You are now *in* a BASH shell running *inside* a container of the custom image for this exercise. From this shell, use your systems administration skills (feel free to Google) to read the contents of `/message.txt` and submit that phrase in the textbox for this assignment.

When you are done in the container type `exit` to exit the shell and stop the container.