**Preparation: Install Docker and VSCode Docker Extension**

If you have a Windows OS version that supports HyperV (Windows 8/10 Professional, Enterprise or Educational) you'll need to follow the instructions at the link below to create a boot profile that turns off HyperV.  If you have Windows 8/10 Home then you can skip this and go on to install Docker Toolbox below.  
You can install and run [**Speccy (Links to an external site.)**](https://www.ccleaner.com/speccy).  This will give you a way to make sure that your system supports virtualization (click on CPU) and what version of Windows you are running (click on "Summary").

[**https://www.hanselman.com/blog/SwitchEasilyBetweenVirtualBoxAndHyperVWithABCDEditBootEntryInWindows81.aspx**](https://www.hanselman.com/blog/SwitchEasilyBetweenVirtualBoxAndHyperVWithABCDEditBootEntryInWindows81.aspx)

The reason for turning off HyperV is that VirtualBox does not work with HyperV.  
  
Install **Docker Toolbox** by following the instructions at: [**Docker Toolbox Installation (Links to an external site.)**](https://docs.docker.com/toolbox/)  
This installation will also install VirtualBox if you don't already have it installed.  If you have VirtualBox installed you don't have to reinstall it during this installation.

You can optionally install the VSCode Docker Extension from: [**VSCode Docker Extension (Links to an external site.)**](https://marketplace.visualstudio.com/items?itemName=PeterJausovec.vscode-docker)  
This will allow you to view Docker containers and images inside of VSCode rather than using the Docker CLI.

After you complete the Docker Toolbox installation, open the "Docker Quickstart Terminal" (look for the icon on desktop).  It will take a bit to initialize (starting Linux VM in VirtualBox) before a prompt is returned.  Once you see a prompt, run the following Docker command in the terminal to ensure that Docker has been installed:

**docker --version**If you bring up VirtualBox, you should see a Linux VM running named "default".You'll use the "Docker Quickstart Terminal" for running the Docker CLI commands in the exercises below and in future course labs.  It runs a Linux Bash shell.  Folder navigation is a bit different than in Windows.  For example, if your files reside in c:\class\lab\week1 then you would enter "cd /c/class/lab/week1" in the terminal to navigate to that folder.

For the exercises below, you'll be working through the lab found at: [**Docker Lab (Links to an external site.)**](https://github.com/docker/labs/tree/master/beginner/)You'll need to create a free account on [**Docker Hub** (Links to an external site.)](http://hub.docker.com/)Note: When the lab mentions "Docker Cloud" or "Docker Store" it is synonymous with what I refer to as "Docker Hub".  
  
Some of the exercises below will ask for a screenshot.  Include all screenshots in a single Word or WordPad file.  Label the screenshots with the exercise and step number.  A good screenshot utility can be found at: [**https://getgreenshot.org/ (Links to an external site.)**](https://getgreenshot.org/) if you don't already have one.

Create a "Week 2" folder for your lab work.  Also, create a subfolder for each exercise this week.Let me know if you have any questions (how to do something, what's happening, etc.) or run into problems.

**Exercise 1 - Hello World (10 points)**

Since you have already installed Docker above you only need to do the following in the "Setup" step of the Docker lab:

Since we are using Docker Toolbox and Not Docker for Windows you do not have to share your drive.

1. Run hello-world. (**screenshot**)
2. Run the following command to list the local images (**screenshot**):  
     
   **docker images**
3. Run the following command to list the containers (**screenshot**):  
     
   **docker ps -a**

**Exercise 2 - Running a Container (20 points)**

Read and work through [1.0 Running your first container (Links to an external site.)](https://github.com/docker/labs/blob/master/beginner/chapters/alpine.md).  T**ake a screenshot after each "docker" CLI command you run.**  Since there are no "steps" just label the screenshots with a sequential number starting from 1.

**Exercise 3 - Sample Web Application in a Container (30 points)**

Read and work through [2.0 Webapps with Docker (Links to an external site.)](https://github.com/docker/labs/blob/master/beginner/chapters/webapps.md).  **Take a screenshot after each "docker" CLI command you run.** Since there are no "steps" just label the screenshots with a sequential number starting from 1.  **Include a screenshot of a browser displaying the output of the Web application.**

**Exercise 4 - Your Web Application in a Container (40 points)**

Repeat 2.3 from [2.0 Webapps with Docker (Links to an external site.)](https://github.com/docker/labs/blob/master/beginner/chapters/webapps.md) with the following exceptions.  **Only include a screenshot of a browser displaying output of your Web application.**

1. Replace the steps in 2.3.1 with the following:

Copy the folders/files from your week1 folder to your week2 folder.

2. Replace the steps in 2.3.2 with the following:

Copy the nginx folder and the Dockerfile from the Week2/app1 folder in [my GitHub repo (Links to an external site.)](https://github.com/oit-gaden/Web-Development-2020-Winter) to your week2 folder.

Proceed with 2.3.3 through 2.3.4.  You'll want to give your Docker images a different name than what you used in Exercise 3 above.

Use port 80 instead of port 5000 in the docker run command since NGINX configuration is configured to listen on port 80 inside the container.  Use port 8080 instead of 8888 in the run command and in the browser to distinguish it from the application in Exercise 3.

**Include a URL to your Docker Hub registry account when you submit your lab. I should be able to pull your image, run it and view your Web application.**

**Exercise 5 (optional) - VSCode Docker Extension**

If you want, install and explore the VSCode Docker extension to see what you can do with it.