Intro to Docker

1 What is Docker?

Docker is a widely-used tools for building and runinng Linux containers:

`https://en.wikipedia.org/wiki/OS-level_virtualization`

A container fulfills many of the same goals as a virtual machine, but is cheaper to run (both in terms of memory and CPU). Whereas a VM provides complete virtual hardware (meaning that you can run any OS on it, and have direct access to devices), a container is simply a funny way of running programs **inside** an existing OS. From the perspective of (most) users, a container is just as good as a virtual machine, but they are far more efficient.

There are multiple container technologies, but in the last few years, Docker has grown to be dominant. While they charge for business users, small users like us can use the software for free. So you should learn Docker now - you will probably be using it at work in the future.

2 Installing Docker

Docker can be installed on Mac, Windows, and Linux. If you are on **Mac or Windows,** go to

`https://www.docker.com`

and download Docker Desktop. We won't be using the Desktop part of things (other than just to get the Docker Engine started each time you reboot), but this is still the easiest way to get Docker installed.

If you are on **Linux,** you will need to Google for information about how to install Docker on your machine. You don't need Docker Desktop at all; if you can install just the command-line tools, that should be sufficient.

2.1 Starting Docker

If you are using Docker Desktop, then you will need to **start** it each time that your machine reboots. This is easy; just open up the GUI once, and wait a couple minutes, until it completes its initialization. Once it is initialized, you can close the GUI; Docker will continue to run in the background.

If you are using only the command-line version of Docker (on Linux), then it shouldn't be necessary to start Docker; it should auto-start when you boot.

3 The Docker Command Line

Once you have installed Docker Desktop and got the engine started, you will be running **all** of your commands on the command line. Open up a Terminal

window (Windows users: Powershell is best) and all of your commands will look like this:

```
docker VERB OPTIONS PARAMETERS
```

For instance, starting a container might be

```
docker run -ti ubuntu
```

On some Linux machines, you may need to prefix every `docker` command with `sudo`, like this:

```
sudo docker run -ti ubuntu
```

4 Using a Container

There are many ways to use Docker, which I encourage you to explore! But for this class, you will generally be running a container that I have already made for you.

252: The container name is `ghcr.io/russ-lewis/csc252-students`

452: The container name is `ghcr.io/russ-lewis/usloss` or `ghcr.io/russ-lewis/usloss-m1` (for M1/M2 Macs)

To run my standard USLOSS Docker image (assuming you already have Docker installed), first `cd` to the directory where you plan to save all of your work for this class. Then, run the following command:

```
docker run -ti -v .:[DIR] [IMAGE]
```

For instance, to run the 252 container, and to place your work in the directory `/root/asm1`, run:

```
docker run -ti -v .:/root/asm1 ghcr.io/russ-lewis/csc252-students
```

Similarly, to run the 452 container, and to place your work in the directory `/root/phase2`, run:

```
docker run -ti -v .:/root/phase2 ghcr.io/russ-lewis/usloss
```

WARNING WARNING WARNING

Docker containers do not, normally, save their contents when they terminate; Docker is a tool that is primarily used for creating ephemeral environments, which are cheap to create and easy to destroy when we're done. But in this case, we're using Docker to **write and test our source code** - and it is of course critical that we not accidentally destroy these files.

This is why we use the `-v` argument above. The `-v` argument is used to "mount" a directory; basically, there will be a special directory, inside the container, which is a mirror of a directory on your laptop. If you write a file on your laptop, in this directory, then it magically becomes visible inside Docker; if you create or update a file inside Docker, this instantly stores the updated file in the directory on your laptop.

The `-v` argument looks like this:

```
-v LAPTOP_DIRECTORY:DOCKER_DIRECTORY
```

If you run the command the way that I suggest above, the laptop directory will be discovered automatically; `cd` into wherever you want to store the solutions to the project, and then run `docker` like I showed you. The command I provided uses the directory `.` (which represents "the current directory"), and passes it to Docker:

```
-v .:DOCKER_DIRECTORY
```

The Docker directory can be anything, but I recommend `/root/phase1` for Phase 1, `/root/asm1` for Asm 1, and so on.

You will notice, when you run the Docker image, that `/root` is kind of the "home directory" for your work, more or less - and that I have already installed USLOSS there.

5 Stopping a Container

A container automatically shuts down when you exit the shell that was running it; you can then start a new container any time that you want, and continue working on your code.

It is perfectly possible to run many containers at the same time; however, I am unsure what would happen if all of those containers mounted the same directory on your laptop, so I don't recommend that you try that.