CSci 4271W (011 and 012 sections) Lab Instructions

Lab 6 **March 3rd, 2025**

Ground Rules. You may choose to complete this lab in a group of up to three students. Before you leave the lab, **make sure you have submitted to Gradescope, you included all group members on the submission, and the autograder found all required files!**

1 Using `docker` for projects, homeworks, and labs

In lecture 12 tomorrow, we will address the topic of isolation, the mechanisms by which an operating system prevents one process (running program) from accessing the resources of another process. Modern operating systems provide a wider variety of isolation concepts beyond the process, including mandatory access controls, sandboxing, containers, and virtualization. Full virtualization creates a "virtual machine" like the ones we are using for the lab, with its own operating system, disk, and so on. Containers also provide a virtual-machine-like abstraction but share a single kernel with other containers and processes on the host, reducing the overhead of running multiple OSes. In this lab, we'll get a feel for Docker, which is the most popular application for creating and managing containers.

2 Install Docker

Log in to your VM, and download and run the installation script for this lab:

```
$ git clone https://github.umn.edu/badlycoded/dockerlab.git
$ cd dockerlab
$ sh ./get-docker.sh
```

This will download and install the docker packages, and configure docker to use the cselabs HTTP proxy. You may be asked if you want to continue the installation with a [Y/n] prompt a few times; alway choose "y". This script also configures docker so that it can be run by an "ordinary" user, by creating and adding users to the "docker" group. To avoid needing to logout and back in, the `newgrp` command starts a shell with the updated group membership:

```
$ newgrp docker
```

Now we should be ready to run docker commands, so try:

```
$ docker run hello-world
```

3 Docker tutorial

Docker has a "workshop" tutorial at https://docs.docker.com/get-started/workshop/. Let's work through Parts 1,2,4 and 5 of this tutorial to install the `todo-app` app as described. A few changes will be required in order to make this tutorial work with the proxy setups on our VMs. Go ahead and start by running the `git clone` command to download the source:

```
$ git clone https://github.com/docker/getting-started-app.git
$ cd getting-started-app
```

This will put you in the correct directory for the remainder of the tutorial.

Second, in order to build the `todo-app` app, we will need to add some lines to the `Dockerfile` created in Step 1 to configure programs that fetch packages from the Internet to use our proxy. Before the `WORKDIR` line, we'll add two extra `ENV` lines, so that the `Dockerfile` should look like this:

```
# syntax=docker/dockerfile:1
FROM node:lts-alpine
ENV http_proxy=http://proxy.oit.umn.edu:3128
ENV https_proxy=http://proxy.oit.umn.edu:3128
WORKDIR /app
COPY . .
RUN yarn install --production
CMD ["node", "src/index.js"]
EXPOSE 3000
```

In Step 2 of "Start an app container", when the tutorial asks you to load the site in a browser, you can use Firefox on the VM with X forwarding. If you haven't already installed Firefox for working on the project, you can run the following in your VM to install `firefox`. (The second command is a persistent version of the workaround for X Windows authentication with snap):

```
$ sudo snap install firefox
$ echo 'export XAUTHORITY=$HOME/.Xauthority' >> $HOME/.bashrc
```

Then run the following from a CSELabs terminal: (Note: this should also work from a personal Mac or Linux machine, but if you're using a Windows box, you may need to use VOLE for this part. You might also prefer VOLE if you're using an off campus network connection.)

```
$ ssh -X student@csel-xsme-s25-csci4271-NNN
student@csel-xsme-s25-csci4271-NNN:~$ firefox &
```

The window that pops up will be a bit laggy, but it'll be a browser running on the VM that can talk to the `todo` container.

Once you've completed Part 2, you can skip Part 3 ("Share the application") as we won't be publishing Docker images in this class.

In Part 4: you're probably not using "Git Bash" (it's a Windows thing) so you can ignore the notes about this section.

In Part 5, you do not need to "Verify that your getting-started-app directory is in a directory defined in Docker Desktop's file sharing setting." (Step 1 under "Trying out Bind Mounts") There's one additional change you'll need to make; replace the command line to start the bind mounted container with the following, analogous to the changes we made to the Dockerfile earlier:

```
$  docker run -dp 3000:3000 \
   -w /app --mount type=bind,src="$(pwd)",target=/app \
   -e "https_proxy=http://proxy.oit.umn.edu:3128" \
   -e "http_proxy=http://proxy.oit.umn.edu:3128" \
   node:18-alpine \
   sh -c "yarn install && yarn run dev"
```

3.1 All done!

Once you've stopped the `node:18-alpine` container, you've finished all the steps the lab! For this lab you'll submit to gradescope the modified Dockerfile you created and the output of `docker image ls`. To create a zipfile with the right contents, cd to your home directory on your vm and run the following commands:

```
$ cp getting-started/app/Dockerfile .
$ docker image ls > docker-images
$ zip dockerlab.zip Dockerfile docker-images
```

Use `scp` to copy the `dockerlab.zip` file back to your lab machine, and then submit the file to the Lab 6 assignment in Gradescope. Make sure you include all of your group members in the submission!

Once you've submitted the file, the autograder will test the to make sure the proper file was submitted, and notify you if it's missing, within a few minutes. If you have the correct dockerfile and images, you'll receive full credit for the lab.

Congratulations, you've finished Lab 6! You might find it interesting and worthwhile to complete steps 6-8 of the workshop, but we're not requiring it for the lab.