```
cd $LABTAINER_DIR/scripts/labtainer-student
```

Then start the lab using the:

```
rebuild [labname]
```

command, where labname is the name of the lab you just created.

The `rebuild` command [5] will remove and recreate the lab containers each time the script is run. And it will rebuild the container images if any of their configuration information has changed. [6] This is often necessary when building and testing new labs, to ensure the new environment does not contain artifacts from previous runs. The progress of the build, and error messages can be viewed in the labtainer.log file. While developing, it is generally a good idea to tail this log in a separate terminal:

```
tail -f labtainer.log
```

If the rebuild fails with a error reflecting a problem resolving hostnames, e.g., mirror.centos.com, please see 14.8.

Note the `rebuild` command is not intended for use by students, they would use the "labtainer" command. The rebuild utility compares file modification dates to Docker image creation dates to determine if a given image needs to be rebuilt. The rebuild may miss file deletions. Thus, if files are deleted, you must force the rebuild using the `-f` option at the end of the rebuild command. Also, addition of symbolic links will not trigger a rebuild. Rebuild references git modify dates (vice file modify dates).

Stop the lab with

```
stoplab
```

When you stop the lab, a path to saved results is displayed. This is the zip file that the student will forward to the instructor.

To test adding a "hello world" program to the new labtainer, perform the following steps:

- From the new lab directory window, cd `$LABTAINER_DIR/labs/[labname]/[labname]`
- Create a "hello world" program, e.g., in python or compiled C.
- From the labtainer-student window, run `rebuild [labname]`

You should see the new program in the container's home directory. If you run the program from the container, and then stop the lab with stoplab, you will see the stdin and stdout results of the program within the saved zip file.

The "hello world" program was placed in `$LABTAINER_DIR/labs/[labname]/[labname]`. The seemingly redundant "labname" directories are a naming convention in which the second directory names one of potentially many containers. In this simple example, the lab has but one container, whose name defaults to the lab name.

The following sections describe how to further alter the lab execution environment seen by the student.

[5] Previously named `rebuild.py`

[6] The build process may generate warnings in red text, some of which are expected. These include an unreferenced "user" variable and the lack of apt-utils if apt-get is used to install packages in Dockerfiles.