3.3 Multiple containers

The `new_lab_setup.py` script can be used to create additional containers for use in the lab. For example, from your new lab directory:

```
new_lab_setup.py -a joe_computer
```

will create a second container for your lab, named "joe_computer". If you again run the rebuild script, you will see two virtual terminals, each connected to one of your two independent computers. Use

```
new_lab_setup.py -h
```

to view the operations available in that script.

The following sections describe how to configure the execution environments on your components, and how to define virtual networks connected to the components.

4 Defining the lab execution environment

A given lab typically requires some set of software packages, and some system configuration, e.g., network settings, and perhaps some lab-specific files. It can include multiple containers, each appearing as distinct computers connected via networks. The execution environment seen by a student when interacting with one of these "computers" is therefore defined by the configuration of the associated container.

Software packages are defined in each container's Dockerfile, described in the subsection below. That is followed by subsection 4.2 describing network definitions, (and other computer attributes) in the start.config file. The remaining subsections then described populating the user HOME directory and system directories, and methods for starting system services and miscellaneous environment settings.

Labtainer containers, by default, present students with a virtual terminal and a bash shell requiring no login. Alternate initial environments, including use of the login program, are described in section 4.7.

Section 4.9 describes how to allow students to share tools they've developed between different labs.

4.1 Docker files

A default Labtainer-specific Dockerfile is placed in the new lab's "Dockerfiles" directory when the new lab is created. And additional Dockerfiles are added when the `new_lab_setup.py -a` script adds computers to the lab. We use standard Docker file syntax, which is described at `https://docs.docker.com/engine/reference/builder/`

Simple labs should be able to use the default Dockerfile copied by the new_lab_setup.py script. That Dockerfile refers to a base Labtainer image that contains the minimum set of Linux packages necessary to host a lab within the framework. The default execution environment builds off of a recent Ubuntu image.

Each container has its own Dockerfile within the

```
$LABTAINER_DIR/labs/[labname]/dockerfiles
```

directory. The naming convention for Dockerfiles is

```
Dockerfile.[labname].[container_name].student
```