# Chapter 10 Lab Questions

During the RHCSA exam, tasks will be presented electronically. Therefore, this book presents most of the labs electronically as well. For more information, see the “Lab Questions” section toward the end of Chapter 10.

## Lab 1

Explore the BusyBox image with Podman. Known for its minimal size, BusyBox includes compact versions of numerous Linux utilities into a single, tiny image, serving as an alternative for a majority of utilities often found in GNU fileutils, shellutils, and so on.

Proceed to pull the image and create a container named busybox by utilizing the image you just fetched. What is the size of the image? Try running the container in detached mode. What is the outcome?

Now, re-create the container, this time launching it with the **/bin/sh** command. Investigate the variety of commands installed within the container image.

Start a container using the busybox image that runs a ping command to www.google.com.

## Lab 2

Set up a simplistic “ping-pong” service running within a container using Python. Start by generating a Containerfile that builds an image based on the Red Hat UBI 9 Python 3.9 image. This image should install the Flask Python module by using the **pip install Flask** command. Moreover, this image is expected to execute a Python script named pingpong.py, comprising the following code:

from flask import Flask
app = Flask(\_\_name\_\_)
@app.route('/ping')
def ping():
 return 'pong'
app.run(host='0.0.0.0', port='8000')

After creating your Containerfile, proceed to build the image and deploy the container in detached mode. Ensure that your application listens to requests on TCP port 1234 of the host machine.

Finally, put the ping-pong service to the test by sending a request to the application. You can do this by running on the host:

$ curl http://127.0.0.1:1234/ping

With this, your Python-based ping-pong service inside a container should respond with “pong.” This confirms the successful creation and running of your containerized Python service.

## Lab 3

Building on what you did in Lab 2, set up the container to run without root privileges using systemd. Name the systemd unit container-pingpong. Make sure to configure the container service to automatically start when the system boots.

As an extra challenge, try doing the same thing, but this time run the service in a container with root privileges.

## Lab 4

In this lab, you will get hands-on experience with running an Apache web service in a container using the UBI 9 httpd-24 image. The exercise focuses on understanding the differences between ephemeral and persistent storage in the context of containerized applications.

First, initiate the UBI 9 httpd-24 container, running it in detached mode while setting it to listen on TCP port 8080.

Next, with the container up and running, connect to it using a shell. In the container environment, create a file called /var/www/html/index.html. You can fill this file with any content you like, such as a welcome message like “Welcome to our webserver!” After creating the file, check to confirm that the web server is properly serving this file by navigating to your localhost’s port 8080.

Subsequently, halt the container and remove it. Then, re-create the container from the same image. Navigate once again to the default web page on your localhost. You’ll notice something peculiar: the file you had created is now missing. As it turns out, this is not a bug but an expected behavior. Any modifications made to a container’s filesystem are ephemeral, meaning they do not persist after the container is removed and are not transferred to the underlying image.

To see how persistent storage works, start by removing the existing container. Next, on your host, create a directory named ~/html and generate an index.html file within it. Re-create the container, but this time use the **-v** option to mount the ~/html directory from your host to the /var/www/html path inside the container. Verify that the web page you have created is now being served correctly.

This time around, if you delete the container and then re-create it while mounting the same volume, the index.html content will persist, illustrating the concept of persistent storage in containerized environments.