Creating a Container Image

Outcomes

1.1 Access a shell prompt and issue commands with correct syntax
1.4 Create and edit text files
1.5 Create, delete, copy, and move files and directories
1.6 Add users, reset passwords, modify user groups, and delete users
1.8 List, set, and change file permissions
2.1 Configure container engines, create, and manage containers
2.2 Create a container image
2.3 Build a container image

Background

In this exercise, you will be helping to create an image for a container that runs the Apache HTTP server on Rocky Linux . The final container should run Apache as the `www` user in the `www` group.

Imagine you are working with a junior sysadmin who is designing a container image to serve web pages. The company you work for has a policy that all web servers must be run by a user named `www` within their environment. Your partner is familiar with Redhat so they have decided to use Rocky Linux as the base for their image. They give you a Dockerfile that isn't quite working and a few tips on what they think needs to be done. Your job is to fix the Dockerfile so that it builds a working image that meets the requirements.

Analysis

This is what they have so far (you'll find this in the class git repo as well):

```
FROM docker.io/rockylinux:9
RUN yum -y update
RUN yum -y install httpd procps
COPY httpd.conf /etc/httpd/conf/
CMD httpd -DFOREGROUND
```

Try building and running this image by executing the following commands within the `create-image` directory:

```
$ docker build -t my-image . ❶
$ docker run -p 8000:8000 my-image ❷
```

❶ Don't forget the `.` here! You're building the Dockerfile *in the current directory*. `-t my-image` tells Docker to tag (name) the image as `my-image`

❷ `-p 8000:8000` forwards port 8000 on the localhost to 8000 on the container

Food for thought…

1. Does this work?
2. Is httpd running as `www`?
3. How could you check?

Modifications

Change the Dockerfile so that Apache runs as `www` and it still works.

Here's some advice:

1. You will have to create a user named `www` and possibly a `www` group. Add RUN commands to the Dockerfile to use the tools we talked about in class.
2. `/var/run/httpd` and `/var/log/httpd` will need to be owned by this user (possibly recursively).
3. The USER command can be used right before `CMD` to make Docker use a different `USER`.
4. Using the syntax in the Analysis section, you should be able to see if httpd is running by going to https://localhost:8000 when your container is running.

Submission

Submit the text of your final Dockerfile to receive credit for this exercise.

Feel free to leave me a comment telling my *why* this is not the best way to solve this problem. *Reading the comments in httpd.conf may point you in the right direction*.