**05 – Docker: Dockerfiles and Images**

**Activities**

COMP190 – Tools and Techniques for Software Development

Dickinson College

Fall 2022

**Name:**

**Introduction:**

We learned a little about Docker in the first class when installing the Docker Desktop application. Since then you have been using Docker Desktop to start and stop the comp190 container containing the Debian Linux environment that we have been using. If you recall, the first step was to fetch the 190-unix-intro *image* from the DockerHub site. You then used that image to create the comp190 *container*, which you have been *starting* and *stopping* using Docker Desktop and connecting to with the Tiger VNC client.

In these activities you will learn how to make your own Docker image and get a feel for how containers help simplify the process of deploying applications. In addition, you’ll gain more insight into the Docker commands you used in HW #1, while also learning how to use the command line to do many of the Docker operations you have been doing in Docker Desktop.

**Installing the Apache2 HTTP Server:**

The motivating example that we will use for this set of activities is the creation of a web server. In this section you’ll setup a web server and create a simple web site on the Debian system we have been using. This will familiarize you with how to install and run the server and where to place the site. Then in a later section you’ll learn how to create a Docker image containing the server and the site.

The web server you will install is called Apache (<https://httpd.apache.org/>), or more fully the *Apache HTTP Server*. Note that HTTP stands for HyperText Transfer Protocol and Hypertext is the language used to create web pages. Apache is a Free and Open Source Software product that has been in use (with continuous improvements) for over 25 years and is one of the most popular web servers used across the web[[1]](#footnote-1).

1. In order to install the Apache HTTP Sever we first need to know the name of the package that contains it.

a. Use the apt search command for: *Apache HTTP Server*. You’ll see that there are multiple packages related to this server. This is common. When you see this, most of the package names, which appear in green, will have suffixes like -bin, -data, -dev etc, or have a prefix of lib (for library). The packages with these prefixes or suffixes are typically dependencies of the main package, or are necessary for development work on the product. Usually (though not always) the package you are looking for to just install and use the product is the one without a prefix or suffix.

What is the name of the name of the package that you think should be used to install the Apache HTTP Server?

b. It is always a good idea to double check that you have the right packages before just launching in and installing stuff. Otherwise, you may end up with a lot of unnecessary software installed. Now use your favorite search engine to search for: *Debian apt install apache*. Scan through a few of the search hits and see if they agree or disagree with your answer to part a.

Based on your review of the search results and part a, what is the name of the name of the package that you now think should be used to install the Apache HTTP Server? If this is different than what you found in a, please explain briefly why your answer changed.

2. From question #1 you should have identified apache2 as the package to be installed. Use the apt to install this package.

a. Give the full command did you used.

b. When the install completes, use the which command to check where the apache2 program is installed. Give a screen shot showing the command that you used and its output.

3. The Apache HTTP Server runs as a server process, thus we will not start it simply by running the program you found in #2.b. Instead, the apache2 package installed a shell script that launches the server for us. Run this script with the following command:

sudo /etc/init.d/apache2 start

Note that start is an argument to this script telling it what we want it to do. Some other arguments that can be used include stop, and restart. Also note that Apache must be run as root so that it has permissions to read and write some of the files and directories that it uses.

Give a screenshot showing the command that you used and the output that it generates. Note there will be an error message displayed that can be safely ignored.

4. I deleted this question but didn’t feel like renumbering everything right now ;)

5. Open the Firefox web browser in your Debian Linux environment at visit the URL:

<http://localhost>

This URL tells the browser to contact the web server (i.e. Apache) that is running on the same machine (i.e. the localhost) and request the default page (i.e. the home page).

You should see the “Apache2 Debian Default Page”. Take a screenshot of your browser window displaying this page and paste it here. If you don’t see this page, then revisit questions #1-#3 to be sure you have installed Apache correctly.

**Creating a new Home Page:**

When a browser contacts a web server using URL, the sever returns the requested web page. For example: The URL https://mysite.org/mypage.html would ask the webserver named mysite.org for the page named mypage.html. If the URL does not specify a page, for example: https://mysite.org or http://localhost then the server will return the default page for the site. This page is often called the *homepage* for the site and is typically stored in a file named index.html. Thus, the URL <http://localhost/index.html> will return the same “Apache 2 Debian Default Page” that you saw before.

6. Put the URL <http://localhost/index.html> into the browser in your Debian Linux environment and confirm that you see the “Apache 2 Debian Default Page” again. Paste a screenshot showing just the part of the browser window with the new URL and the top of the page.

7. The web pages that are served by the Apache server are stored in the /var/www/html directory. The screenshot below shows the long listing (ls -l) for this directory.

a. Give the following information for this directory:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Information** | **Value** |  |
|  | Owner |  |  |
|  | Group |  |  |
|  | Owner’s Permissions |  |  |
|  | Group’s Permissions |  |  |
|  | Other’s Permissions |  |  |
|  |  |  |  |

b. What permissions does your user (comp190) have for this directory?

c. What file(s) are currently in the /var/www/html directory? What permissions does your user (comp190) have for this file?

8. The file that you found in #7.c. should be the same one as described in the paragraph above question #6 that holds the default home page for the web site. So, if you change the contents of that file, then what will be displayed in your browser will change when you visit the site. In this question, you’ll delete that file and create a new one so that the site displays some information unique to you. Be sure to remember what you learned about the permission

a. Delete the index.html file from the /var/www/html. Verify that the file has been deleted. Give a screenshot of the commands that you used and their output.

b. Use nano to create a new file named index.html in the /var/www/html directory. Note: See #7.a - your user will not have permission to edit things in this directory!

Enter the following content into the index.html file, customizing it for yourself, and then save it.

<html>

<head>

<title>My Schedule</title>

</head>

<body>

<h2>My Class Schedule</h2>

<ul>

<li>Put course 1 here (E.g. COMP190)

<li>Put course 2 here.

<li>Add another <li> line for each class.

</ul>

</body>

</html>

Use cat to display the contents of the index.html file. Give a screenshot of your cat command and its output.

c. Use Firefox in your Debian Linux environment to visit the URL:

<http://localhost>

You should see the page that you defined in part b. Take a screenshot of your browser window displaying this page and paste it here. If you don’t see this page, then revisit parts a and b to be sure you have installed Apache correctly.

**Docker Concepts:**

9. Class and homework #01 introduced some of the important Docker concepts. We have reviewed those in class #05 as well. Complete the table below by filling in the appropriate Docker concept from the list below beside the description that is given.

Pull Stop Container Start

Image Repository Create Image Running

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Concept** | **Description** |  |
|  |  | Cloud service where developers share and get Docker image (e.g. Dockerhub) |  |
|  |  | The process of getting an image from an Image Repository such as (Dockerhub). |  |
|  |  | File that contains all of the information about a machine configuration. |  |
|  |  | The process of converting an image into a container. |  |
|  |  | Analogous to an actual Linux machine. |  |
|  |  | The process of getting a container running. |  |
|  |  | A container that is live and can be accessed and used. |  |
|  |  | The process of shutting down a container. |  |
|  |  |  |  |

**Installing Docker in Debian Linux:**

In HW #01 you installed Docker Desktop on your machine so that you could run the 190-unix-intro container containing your Debian Linux environment. In this section, you will install Docker (but not Docker Desktop). In the following sections you will then interact with Docker via the CLI to create a new image containing your web server and site, create a container from your image, run the container and see your site (again). The goal is for you to have a good feel for how Docker images come from and why they are useful.

Docker Install Dependencies:

10. There are a few tools that are used in the install process for Docker for Linux. These will need to be installed prior to beginning the installation.

a. Update apt so that it has the latest package information. What command did you use?

b. Install the package named curl. curl is a program that is used to download files from the web from the command line. You will use it later to get the shell script that installs Docker. What command did you use?

c. Use the which command to verify that curl has been installed. You should see that curl has been installed in the /usr/bin directory. If it has not, revisit the installation command and ensure that you have gotten curl installed before going on. Give a screenshot of your which command and its output.

d. Install the package named apt-utils. This package contains some utility programs that the apt uses to help with configuring other packages when it installs them. One of the utilities installed is the program apt-extracttemplates. Verify that this program has been installed in the /usr/bin directory. If it has not, revisit the installation command and ensure that it is installed before going on. Give a screenshot that shows the program has been installed.

**Install Docker:**

The next step is to install Docker on your Debian Linux machine using the *convenience script*. This convenience script is a shell script provided by Docker that contains all of the instructions necessary for installing Docker so that you do not have to complete them one-by-one. Instead you will download the script using the curl program and then run it.

11. Read through the “Install using the convenience script” section of the following document:

* <https://docs.docker.com/engine/install/debian/#install-using-the-convenience-script>

a. Find the curl command that is used to download the get-docker.sh script. Use that command to download the convenience script. Then use head to show the first 10 lines of the file to ensure that it has downloaded. Give a screenshot of the curl and head commands that you used and their output.

b. Find and run the command that executes the convenience script as the root user. The script will generate a good bit of output as it runs and installs docker.

Nothing is required here. Part c will check that this worked and ask you to revisit the installation if it did not.

c. If you recall, in HW #01, you ran the hello-world example to test if your Docker installation was working. We’ll do that again now. Use the command:

docker run hello-world

Right now, this command should fail with a “permission denied” error - that is expected at this point. If the command does not work, or you get a different error revisit part b and try again.

Paste a screenshot of the docker command you used and the output here.

**Running Docker as a Non-Privileged User (i.e. not as root)**

The output you saw in #11.b shows a “permission denied” error. This indicates that when the docker command was run as your user (comp190) it did not have the permissions it needed to run the hello-world container. One solution would be to run docker as root by using sudo. While that will work, it is not a recommended solution. Running docker as root gives super user privileges to the running containers, which creates a significant security risk. In this section, well fix this issue using what we know about Linux groups and permissions.

12. In the error message in #11.b the “permission denied” error results when docker tries to *connect* to the file /var/run/docker.sock. The docker program reads and writes this file to interact with your host operating system (e.g. MacOS or Windows).

a. Give the following information about the /var/run/docker.sock file.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Information** | **Value** |  |
|  | Owner |  |  |
|  | Group |  |  |
|  | Owner’s Permissions |  |  |
|  | Group’s Permissions |  |  |
|  | Other’s Permissions |  |  |
|  |  |  |  |

b. Compare your (comp190) permissions to for the file /var/run/docker.sock to those of the owner and the group. Based on what you see, what do you think might have caused the “permission denied” error when trying to run the hello-world container as your user?

13. In question #12, you should have seen that the owner (root) and the group (docker) for the file /var/run/docker.sock had write permission but that you (comp190) did not. One way to fix this would be just to give write permission to the “other users.” However, like running docker using sudo, this would create a security concern because all users could then modify this file, which may be undesirable. Thus, the way that non-privileged users are often given permission to run docker is by adding their user to the docker group, which has the appropriate permissions.

a. Use the groups command to check which groups your current user belongs to. Paste a screenshot of the command you used and its output here.

b. The usermod command is used to modify the information about a user on a Unix/Linux system. For example, it can be used to add an existing user to a supplementary (i.e. an additional) group as follows:

usermod --append --groups <groupname> <username>

Use the usermod command to add your user to the docker group. If you get a permission denied error, think about why you might need more permission to complete this operation and how you get that permission.

Give the full command that you used?

c. A Unix/Linux system scans the information about which groups a user is a member of when that user logs in. Thus, it will not automatically recognize when a user has been added to a new supplementary group. To activate your membership in the docker group without logging out and back in you can use the command:

newgrp docker

Run the above command, then use the groups command again to confirm that your user is now in the docker group. Paste a screenshot of the commands that you used and their output here.

14. Now because your user is in the docker group you will have write permission to the /var/run/docker.sock file and thus you should be able to run Docker containers.

Use the command:

docker run hello-world

Take a screenshot of your command and the lines of the input down to where it says “Hello from Docker” and paste it here.

**Building a Docker Image:**

Congratulations you are now able to run other Docker containers within your 190-unix-intro Docker container. Sounds a bit like Inception[[2]](#footnote-2). In this section you will learn how to create your own custom Docker image, use that to create a container and then run that container.

Creating an Image: The Dockerfile:

A *Dockerfile* is a text file containing a script that specifies what is in a Docker image. The commands in a Dockerfile are very similar to those that you use at the command line or include in a shell script. A tool called docker build is then used to run the script in the Dockerfile and generate the image.

15. Create a new directory within your home directory named <name>sServer, where <name> is replaced with your name. Inside that directory create a new text file named Dockerfile with the following contents:

FROM debian:bullseye

RUN apt update

RUN apt install -y apache2

CMD /etc/init.d/apache2 start & sleep infinity

Use cat to display the contents of your Dockerfile. Paste a screenshot of your cat command and its output here.

Quite a bit of what you see in the Dockerfile should look familiar from things you have done in the CLI of your Debian Linux machine. The following text will explain what each of the lines of the Dockerfile does in just a little more detail:

FROM debian:bullseye

The FROM statement specifies the *base image* from which the new image should be built. This way you do not have to start from scratch. You pick a base image that has most of what you want and then add to it. Here we are starting from an image that already contains the bullseye version of Debian Linux. Note: bullseye is just a nickname that Debian has given to version 11, which is the same version you have been using.

RUN apt update

The RUN statement executes any Unix/Linux command and adds the effect of that command to the image. For example, this RUN command ensures that the image contains the most up to date package information.

RUN apt install -y apache2

This RUN statement should look almost familiar, you used a nearly identical command in question #2.a to install the Apache HTTP Server on your Debian Linux machine. This command adds the Apache HTTP Server to the image.

CMD /etc/init.d/apache2 start & sleep infinity

The CMD statement specifies the command that will execute when a container created from the image is started. The first part of the command is exactly the command that you used to start the Apache HTTP Server back in question #3. It ensures that Apache will be running inside the container. The apache2 command is short running and exits once the server is running. That would cause the container to exit as well, and we’d never be able to use it. The & allows us to execute a second command, sleep. The sleep command does nothing, but it waits a specified period of time and then exit (e.g. sleep 5m will wait 5 minutes before exiting). Using infinity as the amount of time ensures that the sleep command will never exit. This will keep the container running, so we can see the web site.

16. The docker build command is used to build an image from a Dockerfile. The general format for this command is:

docker build <context> --tag <image name>

The <context> is the absolute or relative path to the directory containing the Dockerfile.

The --tag argument gives the image that is built a name, called the *tag*, which can be used to refer to the image.

a. Use the docker build command to use your Dockerfile to build an image with the tag <name>-http-img, where <name> is replaced with your name. Give a screenshot of the command you used and the output that is generated.

b. The command docker images will display a list of all of the images that you have locally on your machine, either because they were pulled from Docker hub (e.g. hello-world, 190-unix-intro) or because you built them.

Use the docker images command to ensure that your new image was created. If it was not, revisit questions #15 and #16. Give a screenshot of your docker images command and the output that is generated.

**Creating the Container:**

17. Now that you have built an image for your Apache HTTP Server container, the next step is to create a container from that image. The docker create command is used to create a container from an image. The form of this command that we need is:

docker create --name <container name> --publish 80:80 <image name>

The --name argument specifies the name that will be given to the container.

The <image name> specifies the name of the image from which the container should be created.

The --publish argument is necessary so that we can connect to the Apache HTTP Server inside the container using our web browser. You’ll just have to accept this one, its technical meaning will make more sense after you take a computer networking course.

a. Use the docker create command to create a new container named <name>-http from the image that you built in question #16. Give a screenshot of your docker create command and the output that is generated.

b. The docker ps -a command will display a list of all of the containers that exist on your local system. Use the docker ps -a command to confirm that the container from part a has been created. Hint: Making your terminal wider and the font smaller can make this output easier to read. If the container has not been created, revisit part a and try again. Give a screenshot of your docker ps -a command and the output that is generated.

c. Examine the output from part b. What is the **status** of the container that you created?

**Starting the Container:**

18. You have been using the Docker Desktop GUI to start and stop your 190-unix-intro container. But as with other things we have seen, there are also CLI commands to start and stop containers. The docker start command will start a container (like pressing the “play” button in Docker Desktop). The general form for this command is:

docker start <container name>

a. Use the docker start command to start the container that you created in question #17. Give a screenshot of your docker start command and the output that is generated.

b. Use the docker ps -a command again. Now what is the **status** of your container?

**Viewing the Page:**

The container with the Apache HTTP Server is now running and you should be able to connect to it and request the home page. Earlier you used the URL http://localhost to connect to the server that you installed and directly on your Debian Linux machine. Now, you want to connect to the Apache HTTP Server that is running on the machine inside the Docker container. To do that you will use the *Internet Protocol (IP) address* of that machine, which happens to be 172.17.0.3. There are technical reasons for that, but we won’t worry about them here.

19. Use the Firefox browser in your Debian Linux environment to visit the URL:

<http://172.17.0.3>

This should display the “Apache2 Debian Default Page” that we saw earlier. If it does not, revisit questions #15-#18 to be sure you have the container running. Give a screenshot of part of the Firefox window showing the URL and the top of the “Default Page.”

**Adding the Site:**

If you have spent more than 3 hours on this assignment, skip this section and go to #24.

Notice that the Apache HTTP Server that is running in the container displays the “Apache2 Debian Default Page” not the home page that you created earlier. That is because the container is a completely separate server and you haven’t yet created a home page on that server. Let’s fix that.

20. Create a file named index.html in your <name>sServer directory and save the following content into it.

<html>

<head>

<title>New Page</title>

</head>

<body>

<h2>This is from the container!!</h2>

</body>

</html>

There is nothing required here. Just create the above file.

21. Add the following command on its own line in your Dockerfile between the last RUN and the before the CMD.

COPY ./index.html /var/www/html/index.html

This statement copies the file index.html from the current directory (./) into the container at the location for the homepage (/var/www/html/index.html).

Use cat to display your modified Dockerfile. Paste a screenshot of your cat command and its output here.

22. Use the following commands to stop and delete your <name>-http container and to delete your <name>-http-img image:

docker stop <name>-http

docker rm <name>-http

docker image rm <name>-http-img

Look at the “Containers” and “Images” tabs in your Docker Desktop application to ensure that the container and image have been deleted. Note, you could also use the docker ps -a and docker images commands.

There is nothing required here. Just stop and delete the container and delete the image.

23. Your Dockerfile has now been updated so that it will copy a home page into the container. Use the commands you learned earlier to:

* Build the new version of the <name>-http-img (see #16)
* Create a new <name>-http container (see #17)
* Start the <name>-http container (see #18)
* Display the homepage (see #19)

Give a screenshot of part of the Firefox window showing the URL and the new home page.

**Summary:**

24. Complete the table below by filling in the Linux command that corresponds to each task.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Command** | **Task** |  |
|  |  | Display the groups to which a user belongs. |  |
|  |  | Download a file from the web using the command line. |  |
|  |  | Pause for a specified amout of time. |  |
|  |  | Add a user to a supplementary group. |  |
|  |  | Make a new docker image from a Dockerfile. |  |
|  |  | Make a new docker container from an image. |  |
|  |  | List all of your docker images. |  |
|  |  | List all of your docker containers. |  |
|  |  | Start a docker container. |  |
|  |  |  |  |

**Optional:** To help us improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.

1. <https://w3techs.com/technologies/overview/web_server> [↑](#footnote-ref-1)
2. <https://www.imdb.com/title/tt1375666/> [↑](#footnote-ref-2)