**10 – Docker for FOSS Projects**

**Activities**

COMP190 – Tools and Techniques for Software Development

Dickinson College

Fall 2020

Prof. Grant Braught

As you saw in the previous set of activities just getting a large project to the point where it will build can be a challenge in itself. Managing the configuration and dependencies is challenging even when we were all working on the same operating system. Now imagine trying to manage that with many individuals each installing the project on their own system with its own unique OS versions, configurations and software. The situation quickly becomes extremely challenging. Container based systems (e.g. Docker) provide a modern way of packaging and distributing applications and development environments, ensuring that they are easy to setup and are ready to run anywhere. This set of activities will give you some hands-on experience using Docker that will be helpful in using, developing and managing projects, both FOSS and non-FOSS.

NetworkChuck provides another take on an introduction to docker in his video *You need to learn Docker RIGHT NOW!!*. In addition, he does a worked example that illustrates many of the commands that you’ll be using throughout these activities. While the application he uses is different, the docker commands and their purpose is the same. It is not required viewing, but it may be helpful. You might want to watch this video once now and then again later as the individual commands come up.

* <https://www.youtube.com/watch?v=eGz9DS-aIeY> (23:19)

**Vocabulary:**

One of the biggest challenges faced in getting going with docker is understanding the terminology. The video introduced and gave working definitions of the most important terms. These questions will help you identify and distinguish them.

1. Give a give working definition of the following terms in your own words:

a. Image Repository

b. Image

c. Tag

d. Container

**Preliminaries:**

2. You experienced first-hand in A09 the difficulties of getting setup to build and work on a large software project. One of the advantages of Docker is that it can eliminate the need to install all of the dependencies that are needed for a project. To see what Docker is and how it does this you will start by restoring your Linux Lite to the snapshot that you took in question #10 in A09. This was after you cloned the freeciv repository, but before you installed any of the dependencies that are needed to build it.

a. What was the name of the snapshot you took in A09 question #10?

b. Restore that snapshot and then look at the files in the freeciv directory. Which files are missing that will confirm that you are at a point before you built the project?

c. What happens if you try to build the project? (i.e. try to run autogen.sh and then ./configure).

3. Before you can begin using docker you’ll first have to install it using apt.

a. Install docker using apt. What is the full command that you used?

b. Use the command docker version.

i. What is the version number of docker that was installed?

ii. What error is reported at the end of the output from docker version?

4. Let’s investigate and resolve the permission denied error you saw in the previous exercise. The key to understanding this error is to recognize that the docker version command is attempting to access the file /var/run/docker.sock.

a. Who owns this file? What permissions do they have? (Hint: using ls -l will give the information you need)

b. To which group does this file belong? What permissions do they have?

c. What groups does your user belong to? What command did you use to check? (Hint: Review A05 for the command related to users and groups).

d. Based on your answers to a-c, what permissions does your user have to the file /var/run/docker.sock?

5. Many things that you do with docker will need to interact with this file. So, given the permission denied error, you may at this point be tempted to simply use sudo every time you run a docker command. This will work. However, recall that one of the advantages of using containers is that they are sandboxed.

a. What does it mean that the containers are sandboxed?

b. If you were to use sudo to run docker what user would the docker process run as?

c. What is that user able to do?

6. Based on your answer to the previous question, it should be clear that running docker using sudo would essentially defeat all of the security protections provided by containers. This may not seem bad in your virtual machine, but it can lead to effects that persist when the container is deployed to the cloud. From there, it become vulnerable to attacks from across the Internet. So its best to put good security practices into effect from the start. So instead of using sudo, we will add your user to the docker group. This will give the docker process, launched by your user, permission to read and write the /var/run/docker.sock file without giving it super user privileges.

a. Add your user to the docker group. What command did you use? (Hint: try man adduser or see A05.).

b. Use the command groups to see which groups your user belongs to. Is the docker group listed?

c. Use the command cat /etc/group | grep “<username>”. Is your user listed in the docker group?

d. Hummm… parts b and c should seem to be at odds with each other, showing you conflicting information. The reason is that some system and security settings are initialized when you log in to the system. When those settings are changed it is necessary to log out and back in for them to be reinitialized. Log out and back in. Which groups does your user belong to now?

e. Confirm that docker version now works correctly. What version of the container runtime process (containerd) is being used by the server? Note that the container runtime process is the program running on your machine that manages the containers. This question isn’t expecting you to know what that is, it is just asking something so that I can verify you’ve gotten this working!

**Getting and Running Docker Images:**

One of the big advantages created by docker is that it simplifies the process of establishing a known system configuration. For example, you can pull Docker images from an image repository (e.g Docker Hub) that are pre-configured to include all of the tools and libraries necessary to run or build an application. For FOSS projects this can greatly simplify the process of getting setup to do development. Consider the challenges faced in A09 in getting setup to configure, make and modify FreeCiv. In this section, you’ll get some hands-on experience with using an existing docker image.

7. Docker Hub is “a hosted repository service provided by Docker for finding and sharing container images.” It is a place where developers can upload container images that others can then use to simplify the process of developing or deploying an application.

a. Visit Docker Hub (<https://hub.docker.com/>). Find the official image for Ubuntu. What is the URL for the official Ubuntu docker images page?

b. What is the command shown on this page that will pull the Ubuntu image? Do not use the command, just list it here.

8. The docker pull command you found in the previous question will pull the most recent (i.e. the latest tag) Ubuntu image from Docker Hub. Sometimes this is what you will want. However, the image that you will get will change over time as the developers release new versions. Thus, if you are trying to create a stable configuration you will instead want to pull a specific image (i.e. tag). By doing so you know the exact configuration provided by the image and you know it will be the same every time and for everyone.

a. Using the Docker Hub page for the Ubuntu image, what tags are available for the Ubuntu image?

b. The docker pull command can be modified to pull the image for a specific tag. Visit the Docker Docs (documentation) page for the docker pull command:

* <https://docs.docker.com/engine/reference/commandline/pull/>

What is the *Usage* (i.e. the full syntax) for the docker pull command?

c. Give a command that will pull the Ubuntu image with the 20.04 tag. Recall that in commands (including the Usage statement above, values between [ ] are optional and can be included or omitted and a | indicates that one of the listed options may be included.

d. Use your command from c to pull the Ubuntu image with the 20.04 tag to your machine. What is the full output that is generated when you run this command? The last line should be ﻿docker.io/library/ubuntu:20.04, if it is not, revisit parts b and c and try again.

9. The docker images command will display a list of all of the *docker images* that you have on your machine. This will include not only ones that you have pulled from a repository, but also any that you have created yourself, as we will see later. Use the docker images command to answer the following questions. Note: You may find it helpful to make your terminal wider, or the font smaller, so that the text does not wrap onto multiple lines.

a. The repository and tag fields indicate where the image has come from. What are the values of the repository and tag for the Ubuntu image you pulled?

b. The Image ID is a unique value that you can use to refer to this specific image in other docker commands. What is the ID of your ubuntu:20.04 image? (Notice that in ubuntu:20.04 I am using the repository and tag separated by a colon to identify the image, this is a common practice).

c. How big is the image?

10. The docker ps command is used to list the *docker containers* that have been created on your machine. Find the page for the docker ps command in the docker command line reference (see the column on the left) and use it to answer the following questions:

* <https://docs.docker.com/engine/reference/commandline/docker/>

a. What is the full syntax for the docker ps command?

b. Give a command that will show all containers, both running and not. (Hint: You need to include a command line argument using a - ).

c. Use your command from b. How many containers exist on your machine?

11. The docker create command is used to make a new container from an image.

a. What is the full syntax for the docker create command? What is the URL of the page where you found this syntax?

b. What do the options i and t indicate in a docker create command?

c. Use the command below to create a new container from the ubuntu:20.04 image:

docker create -it --name plain\_ubuntu ubuntu:20.04

Note: Instead of ubuntu:20.04 here you could also use the IMAGE ID.

When the container is created docker outputs the full container ID (a big long hexadecimal number). What is the full container ID for your new container?

d. List all of the containers on your machine now and answer the following questions:

i. In place of the full container ID there is a shorter one that can be used, like the image ID to refer to the container in other docker commands. What is this ID for your container? How is it related to the full container ID?

ii. What is the status of your new container?

iii. The command field indicates the command line that will be issued when the container is started. What command will run when your new container is started?

iv. What name is reported for your newly created container? Why is that the name?

12. The docker start command is used to start (i.e. run) an existing container.

a. Before starting the container open a second terminal in LinuxLite and do the following:

i. Use the whoami command. What does it report?

ii. Use the hostname command. What does it report?

iii. Use the cat /etc/issue command, which shows the linux distribution that is running. What does it report?

b. Now in your original terminal use the following command to start the container you created above:

docker start -i plain\_ubuntu

What text is displayed after you run the above command?

c. What you see in part b is actually a command prompt (i.e. a shell) that is running on the docker container. This is running because the command for the container (see above) was /bin/bash, which told the container to run the bash shell when it starts. Use this shell do the following within the container:

i. Use the whoami command. What does it report?

ii. Use the hostname command. What does it report?

iii. Use the cat /etc/issue command. What does it report?

By comparing part a and c it should be clear that the bash shell running in the docker container is on a different machine that the one in your Linux Lite shell. It has its own hostname, you are a different user (root) and it is even running a different linux distribution (Ubuntu vs LinuxLite). You can think of this container as a hidden machine running inside of your machine. The command line that was started is your way to communicate with this hidden machine. This is one of the big advantages of containers. You can run many different OS versions and configurations within containers on a single machine.

13. Use the terminal on your LinuxLite system (i.e. not the one running in the docker container) to answer the following questions:

a. What is the *status* of your plain\_ubuntu container now? What command did you use to check?

b. Read about the docker stop command in the docker documentation (see earlier link). Give a command that will stop your plain\_ubuntu container.

c. Use the command from b in a terminal in your Linux Lite VM. What happens to the terminal that was running in the container?

d. What is the status of your plain\_ubuntu container now?

14. In this question you will explore the relationship between images and container and the persistence of changes made to containers.

a. Start your plain\_ubuntu container again (Note: you do not need to create the container again, it already exits, just start it). What command did you use?

b. Let’s make some changes inside the container. In the shell running in the container:

i. Change to the root user’s home directory (cd or cd ~). What is the full path to the root user’s home directory?

ii. Try to use the nano editor to create a file. What happens? Resolve this issue. What commands did you use?

iii. Use nano to create and save a simple text file in the root user’s home directory.

c. Stop your container. What command did you use?

d. Create a new container from the ubuntu:20.04 image called another\_ubuntu. What command did you use?

e. Start your another\_ubuntu container. Is nano installed? Is the file you created in the root user’s home directory there?

f. Using a third terminal in LinuxLite, start your plain\_ubuntu container again. Is nano installed? Is the file you created in the root user’s home directory there?

g. Are changes you make persisted (i.e. saved) in the *image* or in the *container*? Briefly justify your answer based on the above questions.

15. As you work with docker, creating and running new containers you can end up with a lot of containers that you just don’t need anymore. The docker rm command provides a way to delete containers that you do not need.

a. Stop both of the containers from the previous question. What commands did you use?

b. Use the docker ps -a command to confirm that the containers have been stopped. What is the status of these containers?

c. Read about the docker rm command in the documentation linked earlier. Give commands that will delete both of your containers. Use the docker ps -a command to check that the containers have been deleted.

d. Does the ubuntu image still exist on your machine after deleting the containers? What command did you use to check?

16. As with containers, when you work with docker you will pull and create new docker images, eventually you will have some that you no longer need and you will want to delete. The docker image rm command is used to delete images. Read about the docker image rm command in the documentation linked earlier.

a. Give a command that will delete the ubuntu image.

b. Use your command from a to delete the ubuntu image. What command did you use to confirm that the image has been deleted?

It is very easy when working with docker to end up with lots of old images and containers hanging around. Each one consumes some disk space on your machine, sometimes significant amounts. So it is worth periodically deleting any unneeded images and containers to reduce the amount of space they consume.

**A Note on docker run:**

If you read about or watch videos on docker or use it with other projects they will often use the docker run command. This command combines the steps of creating the container and starting it. For example, the command:

docker run -it --name plain\_ubuntu ubuntu:20.04

Will both create the plain\_ubuntu container and start it. I have separated them in the above exercises so as not to conflate the two steps. In addition, if you will be starting and stopping containers, you will still need to know about docker start. By learning docker create and docker run separately you now have what you need to know. That said, it is worth being aware of docker run because sometimes it is just convenient and thus lots of people use it.

**Reflection:**

17. Create a list of each of the docker commands that you have used thus far. For each, add a one sentence description of what that command does. This will be a helpful reference as you work though the remainder of the activities.

**Using a Dev Image:**

Now that we have some experience with using a docker image let’s see how it can be helpful to developers. Recall the long process in A09 of installing all of the dependencies that were needed to be able to work on and build FreeCiv. Instead of relying on every new developer to do that kind of setup, a project could instead provide a docker image with all of the development dependencies already installed.

18. I have created just such a FreeCiv development image for us to use. Let’s grab it. Pull the image with the tag 1 from the following DockerHub repository:

<https://hub.docker.com/r/braughtg/fcdev>

What command did you use?

19. Confirm that you now have the image.

i. What command did you use confirm that you have the image?

ii. Give the repository and tag using the repository:tag format mentioned earlier.

iii. What is the Image ID?

iv. What is the size of the image?

20. We can now use that image to create a new container. The command to do so is a bit more complicated than the one we saw earlier for our plan Ubuntu container because we need to do some additional setup.

a. What is the absolute path to the freeciv directory on your LinuxLite machine? (Hint: change to that directory and use pwd).

b. Use the command below to create the new docker container, being sure to replace *<absolute path to your freeciv dir>*﻿ with the path you found in part a above.

docker create -ti \

--device /dev/snd \

-e DISPLAY=$DISPLAY \  
-v /tmp/.X11-unix:/tmp/.X11-unix \

-v *<absolute path to your freeciv dir>*:/home/dev/freeciv \

--name fcdev braughtg/fcdev:1

NOTE: This is one long command. The \ characters allow it to be split across multiple lines to make it easier to read. If you copy and paste the entire command from this document into the terminal in Linux Lite, you will need to remove the blank lines in the “Warning: Unsafe Paste” dialog that comes up (see below). You can edit the command right in the dialog to remove the blank lines. Alternatively, you could copy and paste or type the above command into the Linux Lite terminal one line at a time.

Text

Description automatically generated

21. Now that you have the container, you can start it.

a. Before you do that there is one more bit of configuration you will have to do on LinuxLite. The command below will allow a program running in the container to display a GUI on the LinuxLite desktop. Run this command in a terminal in LinuxLite:

xhost +

What output does this command generate?

b. Start the container. What command did you use?

c. Confirm that the container is running. What command did you use to check? What is the name of the container?

22. Notice that the docker create command above uses the -v option.

a. Using the docker command reference linked earlier, what is the *description* given for the

-v option to the docker create command? Just copy the text that the document gives here. We’ll explore what it means in the next questions.

b. When the -v option is used with two directory paths separated by a colon (:) the first path specifies a directory on the host machine and the second is the path where that directory will appear in the container.

i. Which directory on your Linux Lite machine will be available inside the fcdev container? (See the command in question #20b).

ii. What will the path be to that directory inside the container?

iii. Change to that directory both in the bash shell running in your container and in a Linux Lite terminal. Do these two directories contain the same files?

iv. In your Linux Lite terminal use the command:

touch testfile.txt

Does this file appear in the directory from part ii in the container? How did you check?

v. In the bash shell running in the container use the command:

rm textfile.txt

Is this file also removed from the directory in your Linux Lite terminal?

vi. Based on your answers to parts i-v, give a few sentences in your own words that describe what the -v option to the docker create command does.

23. You can now use the environment provided by the dev container to build the FreeCiv project. Hint: The commands below are the same ones you used in A09 when building FreeCiv so referring to that activity may be helpful. Note however that the container already has all of the dependencies installed, so this time you will not have to install any of them – way easier!

a. Change into the freeciv directory in the dev user’s home directory in the container.

b. Run the command to create the configuration script. What command did you use? (Hint: this is the first step in using the autotools – see A08).

c. Run the configure script. What command did you use? Note: if you get dependency errors you likely ran this in your Linux Lite, which does not have the dependencies installed. Try again I the container.

d. Build the project. What command did you use?

e. Install the project. (Note you will need admin privileges for this to work. The dev user has sudo privileges and the password for dev is dev.) What command did you use?

f. Run the freeciv game! What command did you use?

24. Congratulations, you now have a full development environment setup for FreeCiv using docker. Two big advantages of this approach are:

* By using the docker image, you did not need to go through the process of installing all of the dependencies, as we did in A09.
* By using the -v option to bind mount a volume from the host to the container, you can use all of the tools on your host machine (Linux Lite here) to manage and edit the project code.

To get experience with the second of these you’ll now make an improvement to FreeCiv and then create a pull request to the COMP 190 upstream. There will not be anything to write here for this question. All of your work will be captured in a pull request to the upstream (COMP 190 freeciv) repo at the end.

a. Visit the issues tab of the COMP 190 freeciv repo and read the ticket that is there.

* <https://github.com/dickinson-comp190/freeciv/issues>

b. Run the freeciv client and confirm that the issue exists and is as described.

c. Now imagine that this issue has been assigned to you to fix.

i. Create a new feature branch name dialog-title.

ii. We’ll use grep to identify the files that need to be updated. Using google figure out how to create a grep command that will search all files in a directory and its subdirectories for a specified string. Adapt what you find and give a grep command here that will search in the current directory and all of its sub directories for the title of the dialog that you identified above.

iii. Run your command in the freeciv directory. This will identify all of the files in the project that contain the dialog title to be changed. Notice that there are quite a lot of them. Many are binary files (i.e. compiled and not source code), so we can ignore those, they will be recreated to reflect changes in the source code when you build the project. Many others deal with translations and we’ll ignore those for now also. That leaves us with just the source code files for the C code. Since we have been using the gtk-3.22 client we’ll focus on that one for now, changing the others would be similar. What source code files that are listed are specific to the gtk-3.22 client?

d. You’ve found the files that needs to change, now you need to make the changes. There will not be anything to write here for this question. Your work will be captured by the pull request.

i. Make changes to the files you identified above in b.iii to give the dialog a title that is more consistent with the button the user clicked to get to it. Recall that you can edit the files using an editor in Linux Lite, and because the freeciv directory is mounted in the container the changes will be there automatically.

ii. Build and install the project and run the client to confirm that you have successfully changed the title of the dialog.

iii. Commit your changes to your feature branch, push it to your origin and create a pull request to the upstream for your branch.

**A Note on Translations:**

In the previous question you saw that the title of the dialog box appeared not only in the source code but also in many translation files (e.g. ca.po, pt.po, sv.po, etc…). These .po files contain translations of the phases used in the program. The make process then uses the GNU gettext tool to translate the phrases in the source code into a desired language before compiling them. The gettext tool has helped lots of projects be more inclusive by facilitating the production of executables in a wide variety of languages. It also helps, because it is not necessary to be a programmer to create or edit a .po file. Thus, anyone who would like can help produce translations.

* You can learn more about the gettext library here:
  + <https://www.gnu.org/software/gettext/manual/gettext.html#Why>
* You can learn more about how freeciv handles translations and how you could even contribute to the effort here:
  + <https://freeciv.fandom.com/wiki/Interface_Language>

**Creating a Docker Image:**

This section is optional.

From the above activities you will have gotten a feel for how to use docker and for many of its most important commands. If that has left you wondering where the images on Docker Hub come from and how they are created you might be interested in learning about Dockerfiles. River Lynn Bailey gives a full worked example of creating a Dockerfile, using that to create an image and using that to create a container in the video *Creating your first Dockerfile, image and container*.

* <https://www.youtube.com/watch?v=hnxI-K10auY> (14:45)