Exercise 5: Volumes

In this exercise, we'll learn to work with Docker volumes, for persisting data between containers.

To accomplish this, we'll setup an Apache HTTPD web server, and persist some HTML files in a volume.

Setting up the server

To run our Apache HTTPD server, run this command:

```
alap@sf-cpu-036:~/Exercises-docker$ docker run --rm -d --name apache -p 80:80 httpd:2.4
Unable to find image 'httpd:2.4' locally
2.4: Pulling from library/httpd
26c5c85e47da: Already exists
2d29d3837df5: Pull complete
2483414a5e59: Pull complete
e78016c4ba87: Pull complete
757908175415: Pull complete
Digest: sha256:a182ef2350699f04b8f8e736747104eb273e255e818cd55b6d7aa50a1490ed0c
Status: Downloaded newer image for httpd:2.4
e752023f03c60ab13ecf75237f48863dbc331714f4da0f0fe307761d36b3c125
```

This command will start a new container from HTTP 2.4, name it apache, bind port 80 to the host machine (more on this later), and set a flag to delete the container when it stops.

After it starts, we can run curl localhost to query the web server for the default page:

```
alap@sf-cpu-036:~/Exercises-docker$ curl localhost
<html><body><h1>It works!</h1></body></html>
```

This is the default index.html file included with a new Apache 2.4 installation. Let's replace this HTML file with new content.

To do so, we'll use the docker cp command, similar to scp, which copies files between the host and containers. Let's give it the index.html file as shown below.

```
<html><body><h1>It works in Docker</h1></body></html>
```

The first path is the source path, representing our new file on our host machine, and the second path our destination. apache is the name of the container we want to copy into, and /usr/local/apache2/htdocs/ is where the web server serves HTML from.

Running curl again now looks a little different:

```
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ curl localhost
<html><body><h1>It works in Docker</h1></body></html>
```

A possible data problem

This container, for its lifetime, will continue to serve our new HTML file.

However, containers in Docker are, in practice, considered ephemeral. They can die unexpectedly, and in certain deployments, be removed without warning. If you're depending upon the container state for your application, you might lose important data when such containers die. This is especially a concern for applications like databases, which are supposed to be considered permanent datastores.

In the case of our HTTPD server, simply stopping the container will cause it to be autoremoved. We can bring another container back up in it's place, but it won't have our changes any more.

```
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker stop apache
dockapache
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker run --rm -d --name apache -v httpd_docs:/usr/local/apache2/htdocs -p 80:80 httpd:2.4
4b449f57d7c4d7312ce4b02063a4d451a7785d0a2668e20f8ee41141fff9e3fd
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ curl localhost
<html><body><h1>It works!</h1></body></html>
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$
```

To preserve our data between outages or system upgrades, we can use volumes to persist our data across generations of containers.

Managing volumes

Volumes in Docker are file stores, which sit independently of your Docker containers. The function like Amazon Web Services' EBS volumes, and other mountable media like USB thumb drives. They can be created, deleted, and mounted on containers at specific locations within an image, like you would with mount command in Linux.

To list your volumes, run docker volume ls:

To create a new volume, run docker volume create and give it a volume name.

To remove a volume, run docker volume rm and give it the volume name.

```
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker volume ls
DRIVER    VOLUME NAME
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker volume create myvolume
myvolume
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker volume ls
DRIVER    VOLUME NAME
local     myvolume
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker volume rm myvolume
myvolume
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker volume ls
DRIVER    VOLUME NAME
```

Mounting volumes on containers

First create a new volume named httpd_htdocs:

```
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker volume create httpd_htdocs
httpd_htdocs
```

Then re-run our docker run command, providing the -v mount flag.

```
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker run --rm -d --name apache -v httpd_htdocs:/usr/local/apache2/htdocs -p 80:80 httpd:2.4
fa66c982476d2757b4b4a2806e83dbaeeb22babbb4782055157229d827ce24a4
```

And re-copy in our modified HTML file, and run curl to verify it worked.

```
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker cp ../../mountdir/index.html apache:/usr/local/apache2/htdocs/
Preparing to copy...
Copying to container - 2.048kB
Successfully copied 2.048kB to apache:/usr/local/apache2/htdocs/
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ curl localhost
<html><body><h1>It works in Docker</h1></body></html>
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$
```

Now to see the volume in action, let's stop the container. By providing the --rm flag during run, it should remove the container upon stopping.

```
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker stop apache
apache
```

Then once again start httpd with the same run command as last time. This time, however, we can curl and see our file changes are still there from before.

```
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ docker run --rm -d --name apache -v httpd_docs:/usr/local/apache2/htdocs -p 80:80 httpd:2.4
6da9d65770e17ef546c34b2ae5ed4d318104f7817a3631eafc39d6499b6ea951
alap@sf-cpu-036:~/Exercises-docker/Exercise-5$ curl localhost
<html><body><h1>It works in Docker</h1></body></html>
```

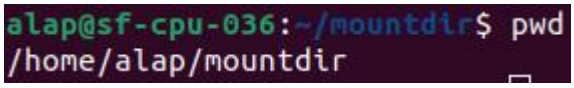
We can take this volume and mount it on any HTTPD container now, which gives us flexibility in swapping out our container for newer versions without losing our data, if we wish.

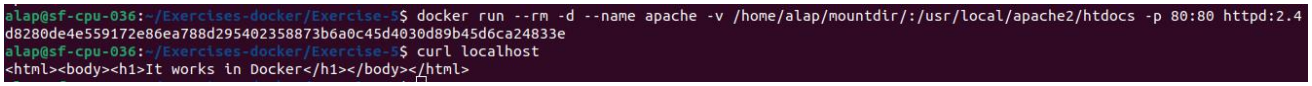
Go ahead and run docker stop apache to stop and remove the container, then docker volume rm httpd_htdocs to remove the volume.

Mounting host directories on containers

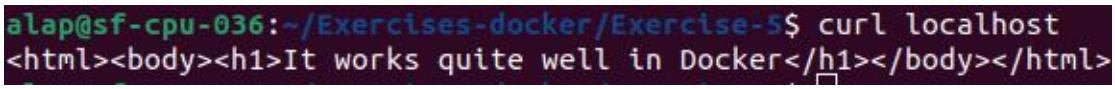
As an alternative to using volumes, if you have a directory on your host machine you'd like to use like a volume, you can mount those too. This is technique is useful in development environments, where you might want to mount your local repo onto a Docker image, and actively modify the contents of a Docker container without rebuilding or copying files to it.

The -v flag to accomplish this is almost identical to the previous one. Simply specify an absolute path to a local directory instead. In our case, we'll pass . to specify the 5-volumes directory in this repo, which conveniently contains a modified version of the HTML file.





With the host directory mount in place, modify the index.html file in this directory with whatever message you like, then save the file and re-run curl.



You can see file changes take place immediately on the Docker container without any need to run docker cp.

Go ahead and run docker stop apache to stop and remove the container.