Lab 1-2 - Doing things with containers

Just being able to get and run images is fairly boring, lets do something a little more useful

Interactive containers

Lets get an image with a basic ubuntu install, we will run the container and then execute a command from inside the container

1. Run Ubuntu interactively

```
$ docker run --rm -it ubuntu /bin/bash

Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
d51af753c3d3: Pull complete
fc878cd0a91c: Pull complete
6154df8ff988: Pull complete
fee5db0ff82f: Pull complete
Digest:
sha256:747d2dbbaaee995098c9792d99bd333c6783ce56150d1b11e333bbceed5c54d7
Status: Downloaded newer image for ubuntu:latest
root@9c4be994e8a7:/#
```

   It looks like we're back to the start, but we're actually inside the Ubuntu container, try the following commands to see that we're not on the system (your hostname will be different)

```
root@9c4be994e8a7:/# hostname
9c4be994e8a7

root@9c4be994e8a7:/# whoami
root

root@9c4be994e8a7:/# cat /etc/issue
Ubuntu 20.04 LTS \n \l
```

   Finish up our session by sending an exit signal (Ctrl+D)

2. Retry those three commands above (hostname, whoami, cat /etc/issue) to see that we're now back in the lab VM

3. We used --rm above, so the container has removed itself upon completion, you should remember how to do this now

```
$ docker container ls -a
```

Disconnected or Daemon containers

In this excersize we'll use a container that will act as a service by continuing to run after we enter the docker run command. We'll use a popular open source web server

1. Run the nginx image in daemon mode, map port 80 in the docker network to the host machine, and we'll give it a name so we can refer to it again in a moment

```
$ docker run -p 80:80 -d --rm --name my_nginx nginx

Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
afb6ec6fdc1c: Pull complete
b90c53a0b692: Pull complete
11fa52a0fdc0: Pull complete
Digest:
sha256:30dfa439718a17baafefadf16c5e7c9d0a1cde97b4fd84f63b69e13513be7097
Status: Downloaded newer image for nginx:latest
70274ee86b47856808ec06007e87b84b47d2642970a28ed15944c514e9967072
```

2. It looks like nothing has happened, but we can now check dockers running processes

```
$ docker ps

CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS               NAMES
70274ee86b47        nginx               "nginx -g 'daemon of…"   25 seconds
ago      Up 23 seconds       0.0.0.0:80->80/tcp   my_nginx
```

and we can check the docker containers (notice that the status for nginx is 'Up' instead of 'Exited')

```
$ docker container ls -a

CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS                      PORTS               NAMES
70274ee86b47        nginx               "nginx -g 'daemon of…"   23 seconds
ago      Up 21 seconds               0.0.0.0:80->80/tcp   my_nginx
ddc075fe365b        hello-world         "/hello"                 16 minutes
ago      Exited (0) 16 minutes ago
interesting_brattain
```

3. Use your web browser to navigate to the IP address you used to connect to the system, you will see the 'Welcome to nginx!' message, so you know the service is still running

4. Lets connect to this running container using the 'docker exec' function, we will use the option -it to connect input and tty output,

```
$ docker container exec -it my_nginx /bin/bash

root@70274ee86b47:/#
```

   We're now looking at a command prompt for the linux system which is running nginx

5. Lets find that file which nginx is showing us

```
root@70274ee86b47:/# cd /usr/share/nginx/html

root@70274ee86b47:/usr/share/nginx/html# ls
50x.html  index.html

root@70274ee86b47:/usr/share/nginx/html# cat index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
...
</html>
```

6. Lets put something of our own in there

```
root@70274ee86b47:/# echo "Hello world from {put your name here}!" >
hello.html
```

   Then navigate to page we just created at

```
http://{ip address of your lab vm server}/hello.html
```

   You should see your message.

7. finally leave the nginx container terminal by sending the exit signal (Ctrl+D), and stop the container

```
$ docker container stop my_nginx
my_nginx

$ docker container ls -a
```

You will see that the container no longer exists (we started it with --rm) because the container gone now and we didn't set up a persistent volume, we've lost our hello world message, if you run step 1. again you will see that it is no longer there.

Conclusion

That is the end of lab 1-2, in this lab you have learned how to interact with running containers, both by connecting input and output to them, and by running as a daemon and then connecting afterwards. We also saw how when containers are removed, the images don't retain any changes we made.