Docker Commit Images

Committing your changes to the container, just like commiting code.

We'll cover the following

In the last lesson, we got access to the shell of the container and were able to run commands. Since we can use the container as a normal Linux machine, let's work in it exactly as we work in any normal Linux machine.

Docker commit

To commit changes to the container and create a new image from it, we need to change something in a container. We will not make this complex to work with. We will simply create a Python program that will print today's date and run it. After that, we will commit the changes and it will create a new image for us, which will have our Python program in it forever.

main.sh

todays_date.py

All code files are copied to end of the page...

> In the first line of the program, we `import` the `datetime` module so that `utcnow()` function can be used. If you are not aware of Python's `datetime` module, it contains a `datetime` file with all the functions in it and print is the module's built-in function.

What is the use of commit? As discussed in the first section, sharing code with someone else with a different environment can lead to bugs.

When you commit the container, a new image is created and you can push that image to the registry. Anybody can fetch the image and will have the same code with a consistent environment. This also helps in deployment as well.

Let's commit the container now. Exit the bash by typing `$exit`.

Then, commit the container by `$docker commit -m "<commit message>" <container_id/name> <new_image_name>:<version>`,

```
(base) adminisatorsmbp:~ venkateshachintalwar$ docker commit -m "Added today's date file" 287ab235
sha256:a9d24607e9d74c58c6469f1c33049e0a8dc459617fe8b40e7d9a866ef9cb705d
```

Now, if you can see the images on your system by typing `$docker images`.

```
(base) adminisatorsmbp:~ venkateshachintalwar$ docker images
REPOSITORY                  TAG                 IMAGE ID          CREATED
             SIZE
date_project                1.0                 a9d24607e9d7    9 seconds ag
o        935MB
```

Next, you should push your image to your Docker Hub account so that anybody can access it.

Steps to push:

- Docker login
- Docker tag <image_name> <your_docker_hub_username>/<image>:<version>
- Docker push <your_docker_hub_username>/<image>:<version>

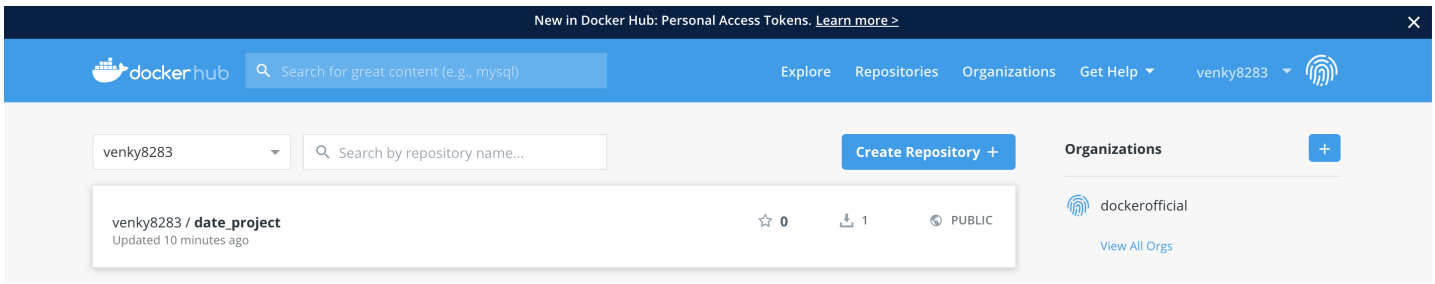
```
(base) adminisatorsmbp:~ venkateshachintalwar$ docker tag date_project:1.0 ven
ky8283/date_project:1.0

(base) adminisatorsmbp:~ venkateshachintalwar$ docker push venky8283/date_proj
ect:1.0
The push refers to repository [docker.io/venky8283/date_project]
81b8d325bd3c: Pushed
ee233c385da7: Pushed
697417af5469: Pushed
536e7dafa51d: Pushed
69e209e74949: Pushed
3bfeb766f97b: Pushed
ea1227feeccb: Pushed
```

```
9cae1895156d: Pushed
52dba9daa22c: Pushed

78c1b9419976: Pushed
1.0: digest: sha256:a7f7f56a77a5d621757a9d53f241a2263a11f919bf92b827d85e939445
8234f9 size: 2429
```

If everything goes well, log into your Docker Hub account and you should be able to see your pushed image.



Ask your friends to run your image. Anyone can pull that image and run it, since it is public.

In the next lesson, we will see how you can connect your host machine's file system with the container's file system.

Code Files Content !!!

⇓ ⇓

```
-----------------------------------------------------------------------
|  main.sh [1]
-----------------------------------------------------------------------


python3 todays_date.py


-----------------------------------------------------------------------
|  todays_date.py [1]
-----------------------------------------------------------------------


from datetime import datetime
print("Today's date is "+ datetime.utcnow().strftime("%Y-%m-%d"))
```

```
**********
```