Docker Run - Accessing Containers

This is the interesting part! Here, we will run the containers from pulled images.

We'll cover the following

Till now, we have only created one container: hello world. Now, we are going to run the containers which can provide the same environment as the host machine or one the application needs. Also, the environment will never change.

Docker Run

If you haven't pulled the Python image in the last chapter, pull it now using `$sudo docker pull python:3.5`. Python is a general-purpose programming language. It is very simple and self-explanatory. We chose Python for demo purposes because any newbie can understand the code just by reading it.

> All the Docker images have the name format of <image name>:<version>. If you do not provide any version number, Docker will consider tag as **:latest** by default.

Commands used in this lesson

- `docker pull \<images-name>:\<version>` : pulls image from Docker registry
- `docker run \<images-name>:\<version>` : runs container from mentioned image
- `docker ps` : shows all running containers
- `docker ps -a` : shows all available containers
- `docker exec` : executes a command in a running container

Let's pull a Python 3.5 image below.

```
(base) Administrators-MacBook-Pro:Downloads venkateshachintalwar$ docker pul
l python:3.5
3.5: Pulling from library/python
4a56a430b2ba: Pull complete
4b5cacb629f5: Pull complete
14408c8d4f9a: Pull complete
ea67eaa7dd42: Pull complete
4d134ac3fe4b: Pull complete
31ae88fffd1d: Pull complete
250ff0d9deb5: Pull complete
2462712d3519: Pull complete
f60ad4796556: Pull complete
Digest: sha256:be81b66442cc6f5ceee040d4347f98a5678a9aab8dc0529f84660ff25e748c5
e
Status: Downloaded newer image for python:3.5
docker.io/library/python:3.5
```

Have you noticed that for the Nginx image, Docker only pulled only three layers, whereas for Python, it pulled more than nine?

Now that we have the Python image, let's run a container from it using `$ docker run python:3.5`

```
(base) Administrators-MacBook-Pro:~ venkateshachintalwar$ docker run python:3.
5
```

If you run this command, you will not see anything, but the container was forked. The container ran and exited as we hadn't asked it to do anything.

`$ docker ps -a` will show us all containers on the system.

```
(base) Administrators-MacBook-Pro:~ venkateshachintalwar$ docker ps -a
CONTAINER ID        IMAGE                  COMMAND                  CREATED
             STATUS                      PORTS               NAMES
b1e466176116        python:3.5             "python3"                2 minutes a
go       Exited (0) 2 minutes ago                       practical_shtern
```

As you can see above, the container was exited. Let's see the meaning of each column in the output.

CONTAINER ID: shows the unique ID of each container

IMAGE: the image from which the container is created

COMMAND: command executed in the container while starting it

CREATED: the time the container was created

STATUS: the current status of the container

PORTS: if any of the container ports is connected to the host machine, it will be displayed here

NAMES: this is the name of a container. If it is not provided while creating the container, Docker provides a unique name by default.

From the beginning of the course, we keep saying Docker uses the Linux kernel for containers. So, let's see that in action.

Type `$docker run -it <container name> <command to execute>`

To get the shell, we will execute bash as a command. The options i & t are used to provide an interactive mode and a pseudo tty terminal.

```
(base) 192:~ venkateshachintalwar$ docker run -it python:3.5 bash
root@287ab2353b27:/# pwd
/
root@287ab2353b27:/# whoami
root
root@287ab2353b27:/#
```

Currently, we are in a virtual Linux OS. It is very lightweight as no extra packages will be installed in it except Python.

Now that you've got the container running, you will be able to access the container's bash. Again, even if you exit the container, it will still be running.

```
root@287ab2353b27:/# exit
exit
(base) 192:~ venkateshachintalwar$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
             STATUS                      PORTS                    NAMES
a5576591262d        python:3.5          "bash"                   10 minute
s ago      Up 10 minutes                                         musing_lic
hterman
```

You can get back to the bash anytime you want by running `$ docker exec -it <container id/name> bash`

```
(base) 192:~ venkateshachintalwar$ docker exec -it a5576591262d bash
root@a5576591262d:/#
```

Now that you know how to get into a container, let's make some changes in the container in the next lesson.