## ##Docker for DevOps Engineers – Day 16

### ## Task:

1. - Use the `docker run` command to start a new container and interact with it through the command line. [Hint: docker run hello-world]
   - **docker run -it --name hello-world -e HELLOWORLD_ROOT_PASSWORD=mypass@123 -d hello-world:latest**

     In above command name should match with repository name from image and deamon name should be same as name. And it will not take password if we will not give ROOT.

     One container id will get generated once we will run docker run command. In our scenario it is "**1253c821bf3c782821c1052561e7642596cc88d07a7504d38da5d56207c52637**" we can see it in the screenshot.

```
ubuntu@ip-172-31-33-42:~$ docker run -it --name hello-world -e HELLOWORLD_ROOT_PASSWORD=Pr@m0d0312 -d hello-world:latest
1253c821bf3c782821c1052561e7642596cc88d07a7504d38da5d56207c52637
```

   - If we will run:
     **docker run hello-world**
     we will get below output

```
ubuntu@ip-172-31-33-42:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

2. - Use the `docker inspect` command to view detailed information about a container or image.
   - docker inspect is a Docker command used to obtain detailed information about a Docker container or image.
   - **docker inspect 48f0efad7b92**


3. - Use the `docker port` command to list the port mappings for a container.
   - **docker port 42c324b42471**
   - **Output:**
     **8001/tcp -> 0.0.0.0:8001**
     **8001/tcp -> [::]:8001**

```
ubuntu@ip-172-31-33-42:~/react_django_demo_app$ docker port 42c324b42471
8001/tcp -> 0.0.0.0:8001
8001/tcp -> [::]:8001
```

4. - Use the `docker stats` command to view resource usage statistics for one or more containers.
   - This command displays real-time usage statistics for running containers.
   - **docker stats**
   - **Output:**

| CONTAINER ID | NAME | CPU % | MEM USAGE / LIMIT | MEM % | NET I/O | BLOCK I/O | PIDS |
|---|---|---|---|---|---|---|---|
| 42c324b42471 | confident_newton | 5.77% | 64.85MiB / 966.2MiB | 6.71% | 1.08kB / 0B | 24.1MB / 782kB | 3 |
| 48f0efad7b92 | Django | 5.74% | 70.68MiB / 966.2MiB | 7.32% | 1.37kB / 0B | 39.8MB / 782kB | 3 |

```
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O         PIDS
42c324b42471   confident_newton  5.78%     64.85MiB / 966.2MiB   6.71%     1.08kB / 0B     24.1MB / 782kB    3
48f0efad7b92   Django            5.91%     70.69MiB / 966.2MiB   7.32%     1.37kB / 0B     39.8MB / 782kB    3
```

   - We can get stats of particular container too by giving cntr_id afted docker stats:
   - **docker stats 42c324b42471**

5. - Use the `docker top` command to view the processes running inside a container.
   - The docker top command is used to display the processes running inside a Docker container. By default, it shows the

processes running in the main process of the container. Here's an example of how to use docker top:

- **docker top 42c324b42471**

```
ubuntu@ip-172-31-33-42:~/react_django_demo_app$ docker top 48f0efad7b92
UID                 PID                 PPID                C                   STIME               TTY                 TIME                CMD
root                4071                4047                0                   06:18               ?                   00:00:00            python man
age.py runserver 0.0.0.0:8001
root                4101                4071                5                   06:18               ?                   00:03:14            /usr/local
/bin/python manage.py runserver 0.0.0.0:8001
ubuntu@ip-172-31-33-42:~/react_django_demo_app$
```

- docker top needs at least one argument.

6. Use the `docker save` command to save an image to a tar archive.
   - The "docker save" command is used to save one or more Docker images to a tar archive. Here's an example of how to use "docker save":
   - **docker save -o docker_image.tar 0a53104ff29e**

```
ubuntu@ip-172-31-33-42:~/react_django_demo_app$ docker save -o docker_image.tar 0a53104ff29e
ubuntu@ip-172-31-33-42:~/react_django_demo_app$ ls
Dockerfile  README.md  api  db.sqlite3  docker-compose.yaml  docker_image.tar  frontend  manage.py  requirements.txt  tests  todo_drf
ubuntu@ip-172-31-33-42:~/react_django_demo_app$
```

   In above image we can see doker_image.tar file is created after we ran docker save command.

   - We can also save multiple images using multiple image_id's:
   - **docker save -o docker_image.tar 0a53104ff29e 3u14514hz15s**

7. Use the `docker load` command to load an image from a tar archive.
   - The docker load command is used to load an image from a tar archive. This command reads the tar archive from the standard input (stdin) or from a specified file and creates an image in the local Docker image registry.

   The syntax for the docker load command is as follows:

   - **docker load -i docker_image.tar**

```
ubuntu@ip-172-31-33-42:~/react_django_demo_app$ docker load -i docker_image.tar
Loaded image ID: sha256:0a53104ff29ebc0d225821a601acd5414b4da6875f0ce8b92d8d1ec52ae4f34d
ubuntu@ip-172-31-33-42:~/react_django_demo_app$
```