Docker for Beginners - Linux

Clone the Lab's GitHub Repo

Use the following command to clone the lab's repo from GitHub (you can click the command or manually type it). This will make a copy of the lab's repo in a new sub-directory called linux_tweet_app.

git clone https://github.com/dockersamples/linux_tweet_app

```
# The PWD team.                                                    #
####################################################################
[node1] (local) root@192.168.0.28 ~
git clone https://github.com/dockersamples/linux_tweet_app
Cloning into 'linux_tweet_app'...
remote: Enumerating objects: 14, done.
remote: Total 14 (delta 0), reused 0 (delta 0), pack-reused 14
Receiving objects: 100% (14/14), 10.76 KiB | 10.76 MiB/s, done.
Resolving deltas: 100% (5/5), done.
```

Task 1: Run some simple Docker containers

There are different ways to use containers. These include:

1. **To run a single task:** This could be a shell script or a custom app.
2. **Interactively:** This connects you to the container similar to the way you SSH into a remote server.
3. **In the background:** For long-running services like websites and databases.

In this section you'll try each of those options and see how Docker manages the workload.

Run a single task in an Alpine Linux container

In this step we're going to start a new container and tell it to run the hostname command. The container will start, execute the hostname command, then exit.

Run the following command in your Linux console.

docker container run alpine hostname

```
[node1] (local) root@192.168.0.28 ~
$ docker container run alpine hostname
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
59bf1c3509f3: Pull complete
Digest: sha256:21a3deaa0d32a8057914f36584b5288d2e5ecc984380bc0118285
Status: Downloaded newer image for alpine:latest
9ff6c033927c
```

The output below shows that the alpine:latest image could not be found locally. When this happens, Docker automatically *pulls* it from Docker Hub.

After the image is pulled, the container's hostname is displayed (888e89a3b36b in the example below).

Docker keeps a container running as long as the process it started inside the container is still running. In this case the hostname process exits as soon as the output is written. This means the container stops. However, Docker doesn't delete resources by default, so the container still exists in the Exited state.

List all containers.

docker container ls --all

```
[node1] (local) root@192.168.0.28 ~
$ docker container ls --all
CONTAINER ID   IMAGE     COMMAND      CREATED              STATUS
           PORTS     NAMES
eb96c75b6ab4   alpine    "hostname"   About a minute ago   Exited (
nute ago             ecstatic_cori
9ff6c033927c   alpine    "hostname"   7 minutes ago        Exited (
ago                  hopeful_wilbur
```

Notice that your Alpine Linux container is in the Exited state.

Run an interactive Ubuntu container

You can run a container based on a different version of Linux than is running on your Docker host.

In the next example, we are going to run an Ubuntu Linux container on top of an Alpine Linux Docker host (Play With Docker uses Alpine Linux for its nodes).

1. Run a Docker container and access its shell.

docker container run --interactive --tty --rm ubuntu bash

```
[node1] (local) root@192.168.0.28 ~
$ docker container run --interactive --tty --rm ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
7b1a6ab2e44d: Pull complete
Digest: sha256:626ffe58f6e7566e00254b638eb7e0f3b11d4da9675088f4781a5
Status: Downloaded newer image for ubuntu:latest
root@2417dfd9bfe5:/#
```

In this example, we're giving Docker three parameters:

- --interactive says you want an interactive session.
- --tty allocates a pseudo-tty.
- --rm tells Docker to go ahead and remove the container when it's done executing.

The first two parameters allow you to interact with the Docker container.

We're also telling the container to run bash as its main process (PID 1).

When the container starts you'll drop into the bash shell with the default prompt root@<container id>:/#. Docker has attached to the shell in the container, relaying input and output between your local session and the shell session in the container.

2. Run the following commands in the container.

ls / will list the contents of the root director in the container, ps aux will show running processes in the container, cat /etc/issue will show which Linux distro the container is running, in this case Ubuntu 20.04.3 LTS.

ls /

ps aux

cat /etc/issue

```
root@2417dfd9bfe5:/# ls /
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@2417dfd9bfe5:/# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COM
root         1  0.0  0.0   4108  3492 pts/0    Ss   16:43   0:00 bas
root        12  0.0  0.0   5896  2876 pts/0    R+   16:54   0:00 ps
root@2417dfd9bfe5:/# cat /etc/issue
Ubuntu 20.04.3 LTS \n \l
```

3. Type exit to leave the shell session. This will terminate the bash process, causing the container to exit.

exit

```
root@2417dfd9bfe5:/# exit
exit
```

> **Note:** As we used the --rm flag when we started the container, Docker removed the container when it stopped. This means if you run another docker container ls --all you won't see the Ubuntu container.

4. For fun, let's check the version of our host VM.

cat /etc/issue

```
[node1] (local) root@192.168.0.28 ~
$ cat /etc/issue
Welcome to Alpine Linux 3.12
Kernel \r on an \m (\l)
```

Run a background MySQL container

Background containers are how you'll run most applications. Here's a simple example using MySQL.

1. **Run a new MySQL container with the following command.**

```
docker container run \
--detach \
--name mydb \
-e MYSQL_ROOT_PASSWORD=my-secret-pw \
mysql:latest
```

```
[node1] (local) root@192.168.0.28 ~
$ docker container run \
>  --detach \
>  --name mydb \
>  -e MYSQL_ROOT_PASSWORD=my-secret-pw \
>  mysql:latest
Unable to find image 'mysql:latest' locally
latest: Pulling from library/mysql
ffbb094f4f9e: Pull complete
df186527fc46: Pull complete
fa362a6aa7bd: Pull complete
5af7cb1a200e: Pull complete
949da226cc6d: Pull complete
bce007079ee9: Pull complete
eab9f076e5a3: Pull complete
8a57a7529e8d: Pull complete
b1ccc6ed6fc7: Pull complete
b4af75e64169: Pull complete
3aed6a9cd681: Pull complete
23390142f76f: Pull complete
Digest: sha256:ff9a288d1ecf4397967989b5d1ec269f7d9042a46fc8bc2c3ae35
```

- --detach will run the container in the background.
- --name will name it **mydb**.
- -e will use an environment variable to specify the root password (NOTE: This should never be done in production).

As the MySQL image was not available locally, Docker automatically pulled it from Docker Hub.

2. **List the running containers.**

docker container ls

Notice your container is running.

```
[node1] (local) root@192.168.0.28 ~
$ docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED
      PORTS                  NAMES
104c896ef738   mysql:latest   "docker-entrypoint.s…"   2 minutes ago
tes   3306/tcp, 33060/tcp   mydb
```

3. **You can check what's happening in your containers by using a couple of built-in Docker commands: docker container logs and docker container top.**

docker container logs mydb

This shows the logs from the MySQL Docker container.

```
[node1] (local) root@192.168.0.28 ~
$ docker container logs mydb
2021-12-15 16:57:56+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Serve
r 8.0.27-1debian10 started.
2021-12-15 16:57:56+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysq
l'
2021-12-15 16:57:56+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Serve
r 8.0.27-1debian10 started.
2021-12-15 16:57:56+00:00 [Note] [Entrypoint]: Initializing database files
2021-12-15T16:57:56.922195Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (my
sqld 8.0.27) initializing of server in progress as process 42
2021-12-15T16:57:56.939898Z 1 [System] [MY-013576] [InnoDB] InnoDB initializatio
n has started.
2021-12-15T16:57:57.758319Z 1 [System] [MY-013577] [InnoDB] InnoDB initializatio
n has ended.
```

Let's look at the processes running inside the container.

docker container top mydb

```
[node1] (local) root@192.168.0.28 ~
$ docker container top mydb
PID                 USER                TIME                COMMAND
7592                999                 0:02                mysqld
```

Although MySQL is running, it is isolated within the container because no network ports have been published to the host. Network traffic cannot reach containers from the host unless ports are explicitly published.

4. **List the MySQL version using docker container exec.**

docker container exec allows you to run a command inside a container. In this example, we'll use docker container exec to run the command-line equivalent of mysql --user=root --password=$MYSQL_ROOT_PASSWORD --version inside our MySQL container.

```
docker exec -it mydb \
mysql --user=root --password=$MYSQL_ROOT_PASSWORD --version
```

You will see the MySQL version number, as well as a handy warning.

```
[node1] (local) root@192.168.0.28 ~
$ docker exec -it mydb \
> mysql --user=root --password=$MYSQL_ROOT_PASSWORD --version
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql  Ver 8.0.27 for Linux on x86_64 (MySQL Community Server - GPL)
```

5. **You can also use docker container exec to connect to a new shell process inside an already-running container. Executing the command below will give you an interactive shell (sh) inside your MySQL container.**

```
docker exec -it mydb sh
```

Notice that your shell prompt has changed. This is because your shell is now connected to the sh process running inside of your container.

```
[node1] (local) root@192.168.0.28 ~
$ docker exec -it mydb sh
#
```

6. **Let's check the version number by running the same command again, only this time from within the new shell session in the container.**

```
mysql --user=root --password=$MYSQL_ROOT_PASSWORD --version
```

```
# mysql --user=root --password=$MYSQL_ROOT_PASSWORD --version
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql  Ver 8.0.27 for Linux on x86_64 (MySQL Community Server - GPL)
#
```

7. **Type exit to leave the interactive shell session.**

```
# exit
[node1] (local) root@192.168.0.28 ~
$
```