MENU ☰

APRIL 15, 2023

DevOps Classroomnotes 15/Apr/2023

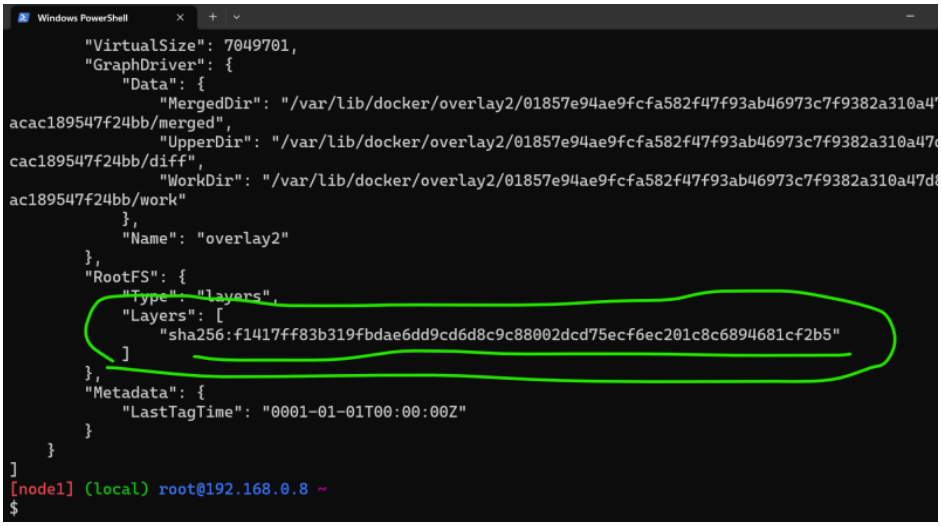
Image Layers

- A Read write layer gets added to every container and image will have read layers

Layers in Docker Image

- Lets pull alpine image and inspect the image

```
docker image pull alpine
docker image inspect alpine
```

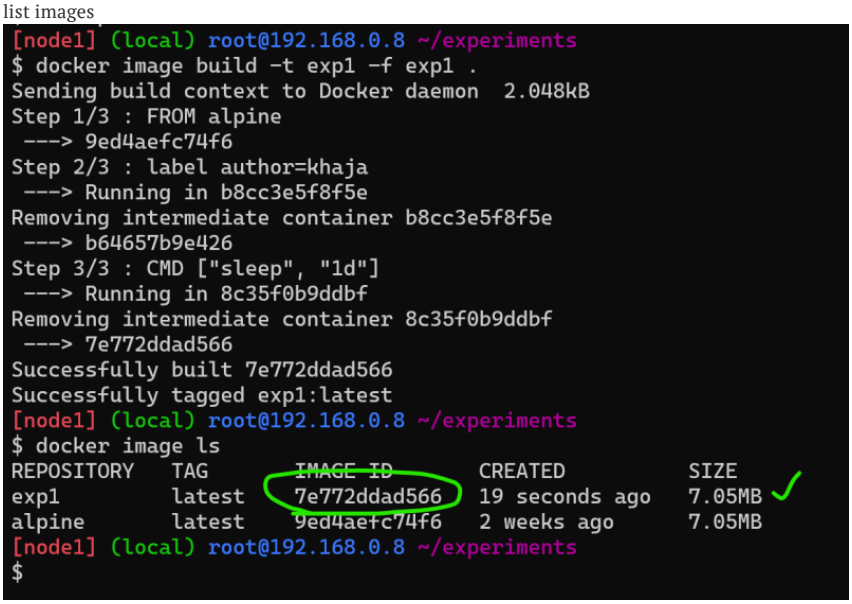


Experiment 1

- Lets create a new image based on alpine `exp1`

```
FROM alpine
label author=khaja
CMD ["sleep", "1d"]
```

- list images



- inspect layers of alpine and exp1

alpine

```
"RootFS": {
    "Type": "layers",
    "Layers": [
        "sha256:f1417ff83b319fbdae6dd9cd6d8c9c88002dcd75ecf6ec201c8c6894681cf2b5"
    ]
},
```

exp1

```
"RootFS": {
    "Type": "layers",
    "Layers": [
        "sha256:f1417ff83b319fbdae6dd9cd6d8c9c88002dcd75ecf6ec201c8c6894681cf2b5"
    ]
},
```

- both have same layers

Experiment 2

- Lets create a new image based on alpine `exp2`

```
FROM alpine
label author=khaja
ADD 1.txt /
CMD ["sleep", "1d"]
```

- lets inspect layers of exp2 and alpine

alpine

```
"RootFS": {
    "Type": "layers",
    "Layers": [
        "sha256:f1417ff83b319fbdae6dd9cd6d8c9c88002dcd75ecf6ec201c8c6894681cf2b5"
    ]
},
```

exp2

```
"RootFS": {
    "Type": "layers",
    "Layers": [
        "sha256:f1417ff83b319fbdae6dd9cd6d8c9c88002dcd75ecf6ec201c8c6894681cf2b5",
        "sha256:8e9773190a318b28ca875f58ac88bfad96baade9e657d76a3df42be1d95514c6"
    ]
}
```

Experiment 3

- Lets create a new image based on alpine `exp3`

```
FROM alpine
label author=khaja
RUN echo "one" > 1.txt
RUN echo "two" > 2.txt
RUN echo "three" > 3.txt
CMD ["sleep", "1d"]
```

- Inspect image layers

alpine

```
"RootFS": {
    "Type": "layers",
    "Layers": [
        "sha256:f1417ff83b319fbdae6dd9cd6d8c9c88002dcd75ecf6ec201c8c6894681cf2b5"
    ]
},
```

exp3

```
"RootFS": {
    "Type": "layers",
    "Layers": [
        "sha256:f1417ff83b319fbdae6dd9cd6d8c9c88002dcd75ecf6ec201c8c6894681cf2b5"
        "sha256:0fe7c6e23b268c83cf665220c13e29acd17c30519cd2457292f257d2ab52276a"
        "sha256:e7766dea9bee1783232ec8fb9e8712f18d5512c6afd7263e1a795d30fc9b45f8"
        "sha256:d0a58e6130636e76b96808e2da45c9e24ba296728aa07bc41354bb583467a1be"
    ]
},
```

Experimenet 4

- Lets create a new image based on alpine `exp4`

```
FROM alpine
label author=khaja
RUN echo "one" > 1.txt && \
    echo "two" > 2.txt && \
    echo "three" > 3.txt
CMD ["sleep", "1d"]
```

- inspect results

alpine

```
"RootFS": {
    "Type": "layers",
    "Layers": [
        "sha256:f1417ff83b319fbdae6dd9cd6d8c9c88002dcd75ecf6ec201c8c6894681cf2b5"
    ]
},
```

exp4

```
"RootFS": {
    "Type": "layers",
    "Layers": [
        "sha256:f1417ff83b319fbdae6dd9cd6d8c9c88002dcd75ecf6ec201c8c6894681cf2b5",
        "sha256:f18f16a7adf032bf9ae76794b8d173e9438e3b208638b4ed0193400393634703"
    ]
},
```

Layers in Docker image contd

- Docker image is collection of layers and some metadata
- Docker image gets first set of layers from base image
- Any Additional changes w.r.t ADD/COPY creates extra layers
- Each RUN instruction which needs some storage creates layer
- It is recommended to use Multiple commands in RUN instruction rather than multiple RUN instructions as this leads to too many layers
- Docker has a filesystem which is aware of layers
    - overlay2

Container and layers

- When a container gets created all the effective read-only image layers are mounted as disk to the container
- Docker creates a thin read write layer for each container.
- Any changes made by container will be stored in this layer
- Problem: when we delete container read write layer will be deleted.
- Refer Here for the article on layers
- Refer Here for layers and storage Drivers

Stateful Appplications and Stateless Applications

- Stateful applications use local storage to store any state
- Stateless applications use external systems (database, blobstorage etc) to store the state
- We need not do anything special if your application is stateless in terms of writable layer, but if it stateful we need to preserve the state.

Solving the Problem with Writable Layers

- Lets create a mysql container Refer Here
- command

```
docker container run -d --name mysqldb -e MYSQL_ROOT_PASSWORD=rootroot -e MYSQL_DATABASE=
```

- To login into container

```
docker container exec -it mysqldb mysql --password=rootroot
```

- To create a table

```
use employees;
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
Insert into Persons Values (1,'test','test', 'test', 'test');
Select * from Persons;
```

```
mysql> Insert into Persons Values (1,'test','test', 'test', 'test');
Query OK, 1 row affected (0.00 sec)

mysql> Select * from Persons;
+----------+----------+-----------+---------+------+
| PersonID | LastName | FirstName | Address | City |
+----------+----------+-----------+---------+------+
|        1 | test     | test      | test    | test |
+----------+----------+-----------+---------+------+
1 row in set (0.00 sec)

mysql>
```

- Now if we remove the container we loose the data
- To fix the problem with data losses, Docker has volumes.
- Volume can exist even after docker container is deleted.
- We can attach volumes to other containers as well
- For this volume to work, we need to know the folder of which data will be preserved
- Let explore docker volume subcommand

```
$ docker volume --help

Usage:  docker volume COMMAND

Manage volumes

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove all unused local volumes
  rm          Remove one or more volumes

Run 'docker volume COMMAND --help' for more information on a command.
[node1] (local) root@192.168.0.8 ~/gol
$
```

- docker volume creates a storage according to the driver specified. The default driver is local i.e. the volume is created in the machine where docker is executing

Docker Volumes

- Refer Here for docker volumes blog

Experiments

- Create a mysql container
- create a postgresql container
- list all the volumes
- inspect all the volumes
- create volume `docker volume create myvol`
- inspect myvol
- Figure out locations of volumes in your local systems

KeyPoints

1. Always ensure volumes are automatically created for the stateful applications as part of Dockerfile (VOLUME instruction)
2. Volumes are of two types
    1. Explicity created (docker volume create myvol)
    2. automatically created as part of container creation
3. Ensure we have knowledge on necessary folders where the data is stored and use volumes for it

Leave a Reply

Enter your comment here...

About continuous learner

devops & cloud enthusiastic learner

VIEW ALL POSTS

PREVIOUS POST

DevOps Classroomnotes 14/Apr/2023