**Image Layers**

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

**Layers in Docker Image**

* Let’s pull alpine image and inspect the image

docker image pull alpine

docker image inspect alpine

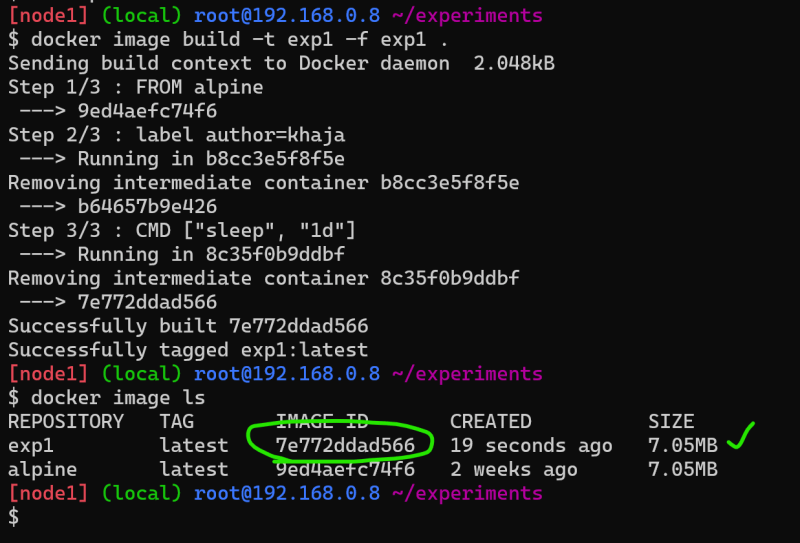


**Experiment 1:** Lets create a new image based on alpine exp1

FROM alpine

label author=khaja

CMD ["sleep", "1d"]



inspect layers of alpine and exp1  


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"]

* let’s inspect layers of exp2 and alpine  
  

**Experiment 3:** Lets create a new image based on alpine exp3

FROM alpine

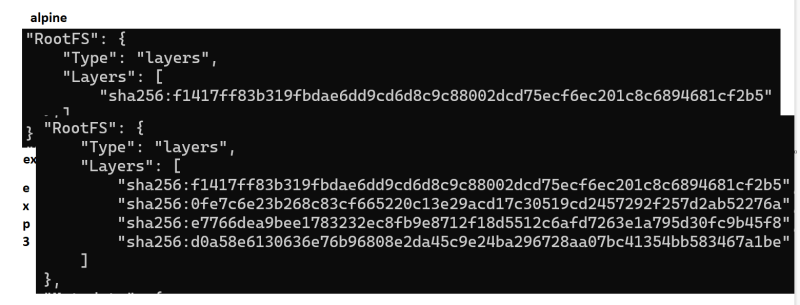
label author=khaja

RUN echo "one" > 1.txt

RUN echo "two" > 2.txt

RUN echo "three" > 3.txt

CMD ["sleep", "1d"]

Inspect image layers  


**Experiment 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  


**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.
* <https://directdevops.blog/2019/09/26/docker-image-creation-and-docker-image-layers/> for the article on layers
* <https://directdevops.blog/2019/09/27/impact-of-image-layers-on-docker-containers-storage-drivers/> 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**

* Let’s create a mysql container <https://hub.docker.com/_/mysql>
* command

docker container run -d --name mysqldb -e MYSQL\_ROOT\_PASSWORD=rootroot -e MYSQL\_DATABASE=employees -e MYSQL\_USER=qtdevops -e MYSQL\_PASSWORD=rootroot -P mysql:8

* 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),

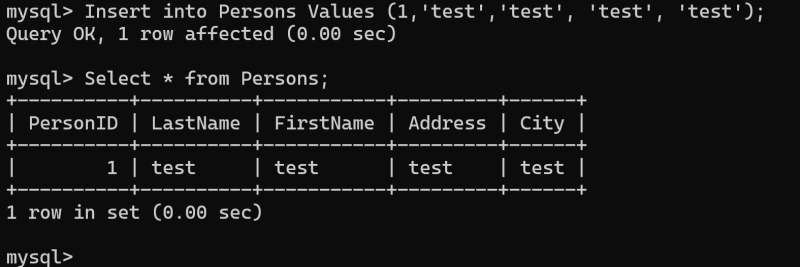
Address varchar(255),

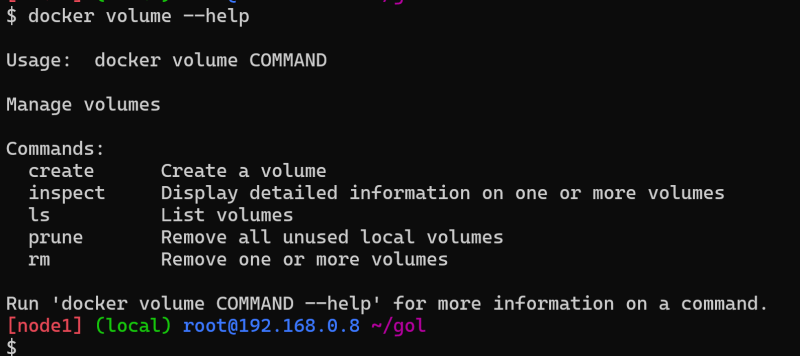
City varchar(255)

);

Insert into Persons Values (1,'test','test', 'test', 'test');

Select \* from Persons;



* 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 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**

* <https://directdevops.blog/2019/10/03/docker-volumes/> for docker volumes blog

**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