Docker-Day-2

22 December 2023 19:34

==

► Docker volumes

==

- Normally the data inside the docker container will be deleted automatically if container stopped/killed.

- In order to make the data that was present inside the container even after the container stopped/killed we use docker volumes.

- Now let's practically see whether data loss will happen or not after container destroy
  Create a container & login

  docker run -it Ubuntu /bin/bash
  echo "Testing the data storage availability" > /storage. out
  cat /storage.out

- Suppose if we kill/exited container the /storage.out file also deleted, but in real-time there will be a situations we need to have file/data present inside the container.

  exit the container & we will not able to see the /storage.out file.

- So docker volumes will help here

- There are different ways of using docker volumes
  - **Anonymous volumes**
  - **Named volumes**
  - **host based volumes**

==

Anonymous volumes

==

- **Create a container a with mount point /data_01 & the data/files present under /data_01 will be accessible on docker host if container destroyed.**
  docker run -it --name myubutnu -v /data_01 ubuntu:latest

- To know **more details about docker containers** we use inspect command,
  docker inspect mybuntu

  - The output of the docker inspect command is in json format.

  - Here in **mount section we find the source tag which present on docker host machine** & pointing to the **/data_01** of docker container.

  - So Inside the docker container under /data_01 if we create any file that will show in docker host on path /var/lib/dcpler/volumes/8491375021/_data

  - And same if we create any files/directory inside docker host machine will be showed in container as well.

- Now kill the container & check whether the data of container is present on docker host or not.

- With this method we will get confused to which volume the container was used if we have many containers.

==

Named volumes

==

- **During named volumes we provide name for the docker volume on host machine**
  docker run -it --name myubutnu -v volume_test:/data_01 ubuntu:latest

- To know more details about docker containers we use inspect command,
  docker inspect mybuntu

  - The output of the docker inspect command is in json format.
    Here in mount section we find the source tag which present on docker host machine & pointing to the /data_01 of docker host.
    We can observe volume name generated as volume_test

  - So Inside the docker container under /data_01 if we create any file that will show in docker host /var/lib/dcpler/volumes/8491375021/_data

  - And same if we create any files/directory inside docker host machine will be showed in container as well.

- Now kill the container & check whether the data of container is present on docker host or not.

- With this method we will get confused to which volume the container was used if we there many containers

==
host based volumes
==

- In this method we will mount specific directory on docker host machine to docker container.

- Create new directory on host machine
  mkdir /root/volume-test

- Mount the new directory(/root/volume-test) as mountpoint(/data_01) inside the container
  docker run -it --name volume_test3 -v /root/volume-test:/data_01 ubuntu:latest

- Create a file inside the docker container on /data_01
  mkdir IamInsideContainer

- On new terminal check the directory "IamInsideContainer" present or not under "/root/volume-test"
  cd /root/volume-test
  ls

- Create new directory on host under /root/volume-test
  cd /root/volume-test
  mkdir IamOutSideContainer

- Verify inside docker container IamOutSideContainer folder present or not under /data_01
  cd /data_01
  ls

- Stop & kill the container
  docker stop <container_id>
  docker rm <container_id>

- Verify the files which are present under /data_01 directory of container are not lost.
  cd /root/volume-test
  ls

==
Port Mapping
==

- Normally every application runs on a specific port.
  Like
    - tomcat will run on port 8080
    - sonarqube will run on port 9000

- Similarly when we spin container for any application it will run particular port,
  We can't directly connect to the docker container via port, so we will bind docker container exposed port with host machine port.
  docker run -it -p 8888:8080 tomcat