Docker Volumes

Manage data in Docker

By default, all files created inside a container are stored on a writable container layer. This means that the data doesn't persist when that container no longer exists.

Docker volumes, which are special directories in a container, store files in the host machine so that the files are persisted even after the container stops.

Volumes are created and managed by Docker. You can create a volume explicitly using the `docker volume create` command.

```
[ec2-user@clarusway ~]$ docker volume create firstvolume
firstvolume
```

When you create a volume, it is stored within a directory on the Docker host. When you mount the volume into a container, this directory is what is mounted into the container. Look at the Mountpoint.

```
[ec2-user@clarusway ~]$ docker volume inspect firstvolume
[
    {
        "CreatedAt": "2020-07-12T13:19:27Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/firstvolume/_data",
        "Name": "firstvolume",
        "Options": {},
        "Scope": "local"
    }
]
```

Declaration of volumes

Volumes can be declared on the command-line, with the --volume or -v flag for docker run. Let's create an alpine container.

```
[ec2-user@clarusway ~]$ docker container run -it -v firstvolume:/sample alpine sh
Unable to find image 'alpine:latest' locally
df20fa9351a1: Pull complete

        Digest: sha256:185518070891758909c9f839cf4ca393ee977ac378609f700f60a771a2dfe
    321 32.77kB/2.798MB
Status: Downloaded newer image for alpine:latest
/ #
```

Tip:

-v or --volume: Consists of three fields, separated by colon characters (:). The fields must be in the correct order.

- the first field is the name of the volume, and is unique on a given host machine. In this example volume name is firstvolume.
- The second field is the path where the file or directory are mounted in the container. In this example folder in container is /sample.
- The third field is optional, and is a comma-separated list of options, such as ro (read only).

Alpine:

- Alpine Linux is an independent, non-commercial, general purpose Linux distribution designed for power users who appreciate security, simplicity and resource efficiency.
- Because of its small size, it is commonly used in containers providing quick boot-up times.

When we type ls command in alpine terminal, we can see the sample folder.

```
[ec2-user@clarusway ~]$ docker container run -it -v firstvolume:/sample alpine sh
Unable to find image 'alpine:latest' locally
df20fa9351a1: Pull complete

            Digest: sha256
    :185518070891758909c9f839cf4ca393ee977ac378609f700f60a771a2dfe321 32.77kB/2
    .798MB
Status: Downloaded newer image for alpine:latest
/ #ls
bin     etc     lib     mnt     proc    run     sbin    sys     usr
dev     home    media   opt     root    sample  srv     tmp     var
/ #
```

We create a file in the sample folder and exit.

```
/ #ls
bin     dev     etc     home    lib     media   mnt     opt     proc    root    run
        sample  sbin    srv     sys     tmp     usr     var
/ # cd sample
/sample # touch file1.txt
/sample # echo "this is added in first container" >> file1.txt
/sample # exit
```

We remove the alpine container.

```
[ec2-user@clarusway ~]$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED
    STATUS                      PORTS               NAMES
2e77f7472339        alpine              "sh"                23 seconds ago
    Exited (0) 17 seconds ago                       intelligent_ellis

[ec2-user@clarusway ~]$ docker container rm intelligent_ellis
intelligent_ellis
```

Let's check the file1.txt.

```
[ec2-user@clarusway ~]$ docker volume inspect firstvolume
[
    {
        "CreatedAt": "2020-07-12T13:36:52Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/firstvolume/_data",
        "Name": "firstvolume",
        "Options": {},
        "Scope": "local"
    }
]
[ec2-user@clarusway ~]$ sudo su
[root@clarusway]# cd /var/lib/docker/volumes/firstvolume/_data
[root@clarusway _data]# cat file1.txt
this is added in first container
```

As we see above, file1.txt is still there even if we remove the container.

Usage volume with different containers

Let's run an alpine image and this time we will create try1 folder instead of sample folder.

```
[ec2-user@clarusway ~]$ docker container run -it -v firstvolume:/try1 alpine sh
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
df20fa9351a1: Pull complete

            Digest: sha256
    :185518070891758909c9f839cf4ca393ee977ac378609f700f60a771a2dfe321
Status: Downloaded newer image for alpine:latest
/ # ls
bin    dev    etc    home   lib    media  mnt    opt    proc   root   run    sbin
       srv    sys    tmp    try1   usr    var
/ # cd try1
/try1 # ls
file1.txt
/try1 # cat file1.txt
this is added in first container
```

As we see, we can reach file1.txt via a new container.

We can add a new file to the try1 folder.

```
/try1 # touch file2.txt
/try1 # echo "this is added in second container" >> file2.txt
/try1 # cat file2.txt
this is added in second container
/try1 #
```

We create an ubuntu image.

```
[ec2-user@clarusway ~]$ docker container run -it -v firstvolume:/try2 ubuntu sh
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
692c352adcf2: Pull complete
97058a342707: Pull complete
2821b8e766f4: Pull complete
4e643cc37772: Pull complete
Digest: sha256:55cd38b70425947db71112eb5dddfa3aa3e3ce307754a3df2269069d2278ce47
Status: Downloaded newer image for ubuntu:latest
# ls
bin  boot  dev  etc  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root
     run  sbin  srv  sys  tmp  try2  usr var
# cd try2
# ls
file1.txt  file2.txt
```

We can use the same volumes with different containers.

Use a read-only volume

For some development applications, the container needs to write into the bind mount so that changes are propagated back to the Docker host. At other times, the container only needs read access to the data. Remember that multiple containers can mount the same volume, and it can be mounted read-write for some of them and read-only for others, at the same time.

```
[ec2-user@clarusway ~]$ docker container run -it -v firstvolume:/try3:ro
    centos sh
Unable to find image 'centos:latest' locally
latest: Pulling from library/centos
6910e5a164f7: Pull complete

                        Digest: sha256
    :4062bbdd1bb0801b0aa38e0f83dece70fb7a5e9bce223423a68de2d8b784b43b
Status: Downloaded newer image for centos:latest
sh-4.4# ls
bin  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root
     run  sbin  srv  sys  tmp  try3  usr  var
sh-4.4# cd try3
sh-4.4# ls
file1.txt  file2.txt
```

Let's try to add a file to the volume.

```
sh-4.4# touch file3
touch: cannot touch 'file3': Read-only file system
sh-4.4#
```

Complementary Lesson about Docker Volumes