Volumes are basically directories of host system managed by Docker
There are three main use cases for Docker data volumes:

1. To keep data around when a container is removed
2. To share data between the host filesystem and the Docker container
3. To share data with other Docker containers

Docker run has a -V flag which allows to set data volumes that are mounted inside of our container

1. Run the docker container by mapping the current directory as /mist in the container. We are adding a local drive as a volume to the contianer

```
[puppet@root$:/test/Master]$  docker run -it -v $(pwd):/mist ubuntu bash
root@dea9f1c0a043:/# df -hT
Filesystem Type   Size  Used Avail Use% Mounted on
tmpfs      tmpfs  3.8G     0  3.8G   0% /dev
tmpfs      tmpfs  3.8G     0  3.8G   0% /sys/fs/cgroup
/dev/sda2  xfs     49G   12G   38G  24% /mist
```

Now once container is started we can check the file system using “df -hT” which will show the /mist as a drive attached.

Create a test file in the /mist location

```
root@dea9f1c0a043:/# cd mist/
root@dea9f1c0a043:/mist# echo "hello World" >> hai.txt
root@dea9f1c0a043:/mist# cat hai.txt
hello World
root@dea9f1c0a043:/mist# exit
exit
```

Once we exit ,we can see the created file in the container in our local repository.

```
[puppet@root$:/test/Master]$  cat hai.txt
hello World
```

The Mounts are read/write. But docker allows us to create a read only mount

2. In the above example we have seen how we can share a local location to a the container as a volume. Docker also provides another way to map volumes of a container to another container using the --volumes-from argument. This can be run as

```
docker run -it --volumes-from 0ff56994a111 ubuntu bash
```

The above command will add the volumes attached to the container 0ff56994a111 to the new container that will be created from above command.