[**Docker Persistent Storage**](http://www.bogotobogo.com/DevOps/Docker/Docker_Container_Persistent_Storage_Data_Share.php)

**Communication among Docker images**

By default the host machine has no access to the files inside the Docker container because Docker containers are isolated from the host machine by default.

However, we have three couple of ways to get around this.

1. The most common way is to have Docker specify environment variables inside the Docker container. Then, he code running inside the Docker container will check the environment variables on startup and use them to configure itself accordingly.
2. Another commonly used method is a Docker data volume as we'll see in subsequent sections of this page.
3. The last way of communicating with a Docker container is using **links** or port forwarding.

**Data volumes**

Data volumes are designed to persist data, independent of the container's life cycle.

Docker therefore never automatically delete volumes when you remove a container, nor will it "garbage collect" volumes that are no longer referenced by a container.

**Adding a data volume**

Though we executed docker run commands with various argument combinations in the previous chapter([More on docker run command (docker run -it, docker run --rm, etc.)](http://www.bogotobogo.com/DevOps/Docker/Docker_Run_Command.php)), docker run was not doing useful operations.

So, in this chapter, we'll learn more about docker run commands that doing more useful things.

We're going to run docker run command with -v argument:

k@laptop:~$ docker run -it --name yaong2 -v /mydata ubuntu:latest /bin/bash

Let's create files under the **/mydata** folder:

root@2729ea1e66b9:/# cd /mydata

root@2729ea1e66b9:/mydata# ls

root@2729ea1e66b9:/mydata# touch a b c

Get out of the container (Ctrl + P + Q) without stopping it.

**Locating a volume**

We can locate the volume on the host by utilizing the "docker inspect" command. The output will provide details on the container configurations including the volumes:

k@laptop:~$ docker inspect yaong2

[{

...

"Volumes": {

"/mydata": "/var/lib/docker/vfs/dir/0b385b8fcc9c5c68f0ab80f7acda9d6582bb5b7dbc9307e75b3eafed6bf118df"

},

...

}

]

Let's check if the files we created on the container are really in the host file system:

k@laptop:~$ sudo su -

root@laptop:~# ls /var/lib/docker/vfs/dir/0b385b8fcc9c5c68f0ab80f7acda9d6582bb5b7dbc9307e75b3eafed6bf118df

a b c

Yes, the files 'a','b', and 'c' are there in our host machine!

**Mount a host directory as a data volume**

We can mount a directory from our Docker host into a container.

We have three files 'a', 'b', and 'c' in **host-volume** folder:

k@laptop:~$ ls host-volume

a b c

Now we want to mount the directory to our Docker container:

$ docker run -it --name=yaong3 -v /home/k/host-volume:/opt/host-volume ubuntu:latest /bin/bash

This will mount the host directory, **/home/k/host-volume**, into the container at **/opt/host-volume**.We can check if our container is sharing the volume:

root@657e3a50bf59:/# ls /opt/host-volume

a b c

Let's remove the 'c' file and check the host file system if it's really been removed:

root@8d87eb4aa802:/# rm /opt/host-volume/c

root@8d87eb4aa802:/# ls /opt/host-volume

a b

Type "Ctrl + P + Q" to exit without stopping the container:

k@laptop:~/host-volume$ ls /home/k/host-volume

a b

Yes, we can see clearly the file sharing is working!

Stop the container:

$ docker stop yaong3

yaong3