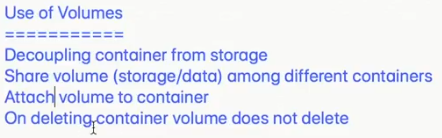
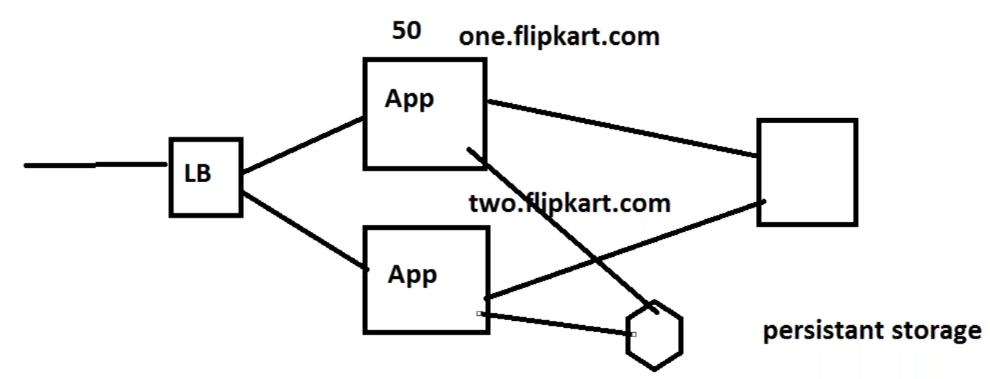
* If we don’t specify any location to store the data of container. It automatically stores the data inside the container and once the container is removed, the data is lost



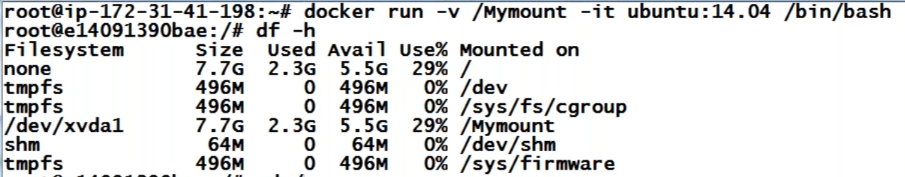
**Volumes:**

* If we increase our system configuration, it is called vertical scaling
* If we add more servers for the work load is called horizontal scaling. This is preferred by most of the companies now
* Whether the workload goes to first or second server will be managed by load balancer. It identifies and transfers the traffic
* Whatever the transactions happen in application will be stored in database



* This persistent storage needs to be present in the docker daemon server
* If the container gets killed, we should not loose the data, this will be managed by using docker volumes
* Docker volume can be created in the docker file or we can pass the arguments while creating the container
* Persistent storage will be there even after the application killed





* It will be shown as a mount point in container, but it actually stores the data in physical machine
* Go to /mymount in container and create one file

By using the inspect command with container id, we can find the source path as below, go to that path, we can find the file which we created in container

* **Docker inspect <CID or Image ID or name> 🡪 for the details**

We can also inspect with volume name

* **Docker inspect <volume name> or docker volume inspect <volume name>**
* We can see the source path under mount section
* Source path is our physical machine location and destination is in container



* Now, enter some data in that file and we can see the same data inside the container file also

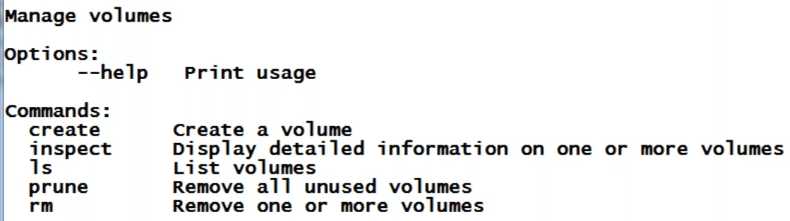


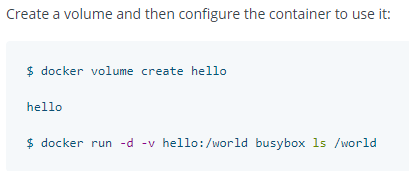
* Where ever we write, it will show in both the machines

These two are always in sync

* **Docker volume ls 🡪 to check the volumes**
* **From here we cannot find the volume ids. For more details we need to use inspect command**

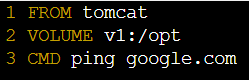
Basically, we can create the volumes and use them as mount points also inside container



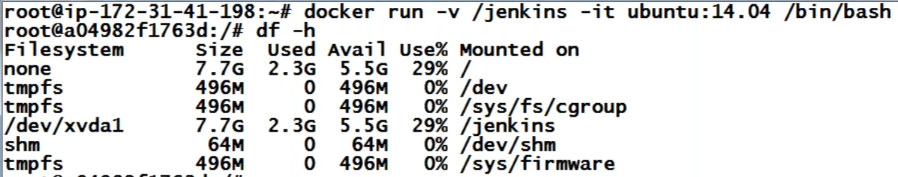








* Now, let’s kill the container and run it with different mount point as below

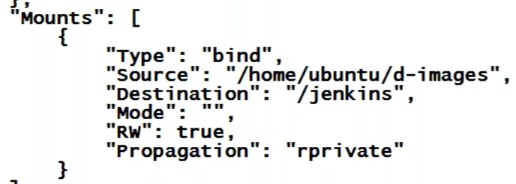


* We need to always find out the volumes for the docker container with inspect command
* So, we can use the below alternative method, we can map our directory to the mount point
* Here, the volume would be present at the place where ever we want



**As above, we can give any directory as volume inside physical server. These are called bind mount**

* Now, we can see the volume details with inspect as below



* Even if the container is present in ps-a, those volumes will not be removed



* Here, we mapped Jenkins\_home to the volume /Jenkins
* And this mount can be shared with multiple containers
* Stateless applications don’t store anything on the containers, it stores on DB. It is not required volumes
* For stateful, we need to have volumes
* Containers have data only on volumes
* These are the two main advantages of volumes, one is we can have persistent data. So that it will be secure if container fails
* And another is we can have multiple containers use same volumes, the mount points in the containers need not to be same

**Multiple volumes for a single container:**

**If we have given one volume in the image and try to use another using -v in command, docker will create container with the volume which was given in command**

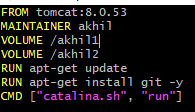
**When we map a directory as a volume to the container. The data inside the directory will be reflected on to the mount point inside container**

**When we commit the container to an image, it will take the volume details as well**

**We can not add two directories to the same mount point as below**

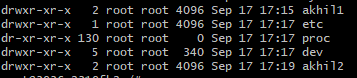


* We can create two volumes as below ways





* We can see the volume mount points at / mount point as below inside the container



* As above, the data gets sync between akhil1 to akhil1 and akhil2 to akhil2



* When we are creating two mount points pointing to same directory. Only first mount point is getting created we can find both in / mount point only
* Both are able to get sync with same directory
* We cannot rename the volumes

**Docker volumes drive as local means the docker volumes are created on the same host where container is running**

**Bind mount:**

* Bind mount is also same as volume whereas here we can use any directory to mount on container
* We can find the bind mount details by inspecting container