**Docker Tutorial 3 - Demystifying Volumes**

**setup**

FROM ubuntu:14.04

MAINTAINER John Willis <john@socketplane.io>

VOLUME ["/john99"]

CMD ["/bin/sh"]

docker build -f myimage.dockerfile -t myimage .

**Volumes**

(***Simple Run***)

Dynamically create file system

Create as part of image // available to all containers based on the image

docker run -it -v /john1 busybox

cd john1

touch file1

ctrl-p-q # Keep in running

docker restart <cid>

docker exec <cid> ls /john1

The file stays until we remove the container  
however, if we start a new container based on the orgi busybox image

docker run -it -v /john1 busybox

cd john1

ls (not there) This was a new run..

exit

We can also have volumes that have volumes defined in the image build..

docker images (show my image)

docker inspect <imageid>

docker history <imagename>

docker run -itd myimage

cd mydir

ls (file1 will alwys be there)

touch file2 (same rues apply)

ls

exit

docker run -it -v /john1 myimage

ls (both directories)

cd into bothâ€¦

on docker host..

mkdir john3

cd john3

touch file3

touch file4

docker run -it -v /vagrant/john3:/john3 myimage (host needs to be abs path)

**good for testing src codeâ€¦**

cd john3

ls

touch file5

ls

exit

on docker host..

cd john3

ls (see files 3 and 4 and 5)

Read ONLY : that is u cant write to the file system

docker run -it -v ~/john3:/john3:ro myimage (point out ~)

cd john3

vi file5

save???

docker run -it -v ~/.bash\_history:/.bash\_history myimage

docker ps -a (see what's running)

docker kill $(docker ps -q)

docker rm $(docker ps -aq)

docker ps -a

docker run -it --name john1 -v ~/john3:/john3 myimage

ls (directories are there.. and files trust me)

### ctrl-pq (keep running)

docker ps

docker run -it --name john2 --volumes-from john1 myimage

ls

### ctrlpq

docker run -it --name john3 --volumes-from john2 myimage

\*\*\*(make a backup)\*\*

docker run --rm --volumes-from dbstore -v $(pwd):/backup ubuntu tar cvf /backup/backup.tar /dbdata

Here you’ve launched a new container and mounted the volume from the dbstore container. You’ve then mounted a local host directory as /backup. Finally, you’ve passed a command that uses tar to backup the contents of the dbdata volume to a backup.tar file inside our /backup directory. When the command completes and the container stops we’ll be left with a backup of our dbdatavolume.

You could then restore it to the same container, or another that you’ve made elsewhere. Create a new container.

$ docker run -v /dbdata --name dbstore2 ubuntu /bin/bash

docker run -it --name vol-test -h CONTAINER -v /data ubuntu /bin/bash

where to find it in host

docker volume ls

to write to the file from outside container

touch /var/lib/docker/volumes/8e0b3a9d5c544b63a0bbaa788a250e6f4592d81c089f9221108379fd7e5ed017/\_data/test-file

create volumes using the docker volume create command:

docker volume create --name my-vol

docker run -d -v my-vol:/data ubuntu

this will mount the my-vol volume at /data inside the container.

Two ways to share volumes between containers

Container to container and from location where first container created

docker run -it -h NEWCONTAINER -v 8e0b3a9d5c544b63a0bbaa788a250e6f4592d81c089f9221108379fd7e5ed017:/my-data debian /bin/bas

A volume will never be deleted as long as a container is linked to it

### Deleting Volumes

Volumes are only automatically deleted if the parent container is removed with the docker rm –v

 a volume will only be deleted if no other container links to it. Volumes linked to user specified host directories are never deleted by docker.

To delete all volumes not in use, try:

docker volume rm $(docker volume ls -q)