Docker Certified Associate

Docker Storage Layers

Images and Layers

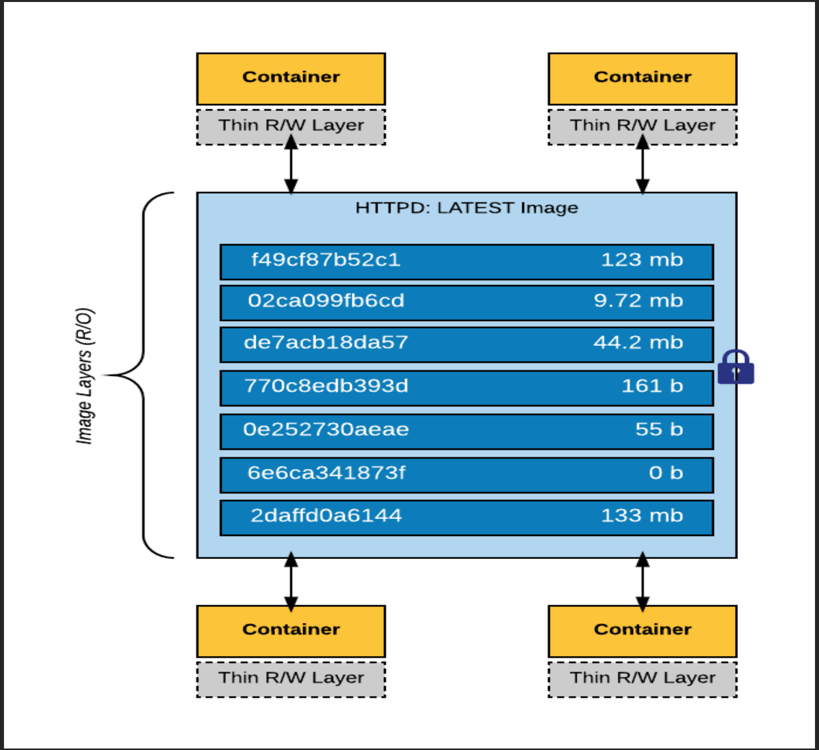
A Docker image is built up, from a series of layers, each representing a single instruction in the image’s Dockerfile. Every layer except for the last, is a read-only layer.

A Dockerfile like:

```
 FROM centos:latest
RUN yum update -y
RUN yum install -y telnet
ENTRYPOINT echo “This container has finished”
```

Would create four discrete layers in the construction of the image. Some commands can be chained so that multiple commands actually run, but only build a single layer (‘RUN yum update –y && yum install –y telnet’ for example).

Images and Layers



In the case of the 'httpd:latest' image depicted to the left from Docker Hub, the image consists of 7 discrete layers.

Each layer represents a command that was run in the assembly of that image. Each layer is only the deltas from the previous image laying on top.

The only writable layer then becomes the 'Thin R/W Layer' for each container instantiated on that image's R/O layers.

Images and Layers

The configured Docker Storage Driver handles the details about how the image layers interact. See more information on Storage Driver Use Cases to determine which is supported and optimal for your platform, distribution and work load.

With each container having its own writable container layer, each image can maintain a 1 to N ratio of image storage to container storage (i.e. the image storage layers are never repeated).

As a result of Docker's storage strategy of 'Copy-On-Write', files and directories that exist in lower layers that are needed in higher layers, can be provided read access to them to avoid duplication. If that file needs to be modified, only then is it copied to the higher layer where the changes are stored.

Containers and Deletion

Any time a container is deleted, any data that is written to the container read/write layer that is NOT stored in a data volume, will be deleted along with the container.

Data volumes, since they are NOT controlled by the storage driver (since they represent a file or directory on the host filesystem in the /var/lib/docker directory), are able to bypass the storage driver. As a result, their contents are not affected when a container is removed.

This allows containers to remain portable while providing a method of persistent storage outside of the image and container layered filesystem structure.