Chapter 7 - Volumes

Volume Mounts

Volume mounts are for sharing data between multiple containers, either in parallel or on sequential executions. Now let's run an example:

Create a volume:

```
docker volume create my-vol
```

Check that it exists:

```
docker volume ls
```

Start a container that uses the volume:

```
docker run -it -v my-vol:/app alpine
```
The `-v` flag is used to specify a mount, in the format `<volume name>:<mount point>`

Once in the shell, create a file:

```
touch /app/index.js
```

Drop out of the container:

```
exit
```

Start a new container, using the same volume:

```
docker run -it -v my-vol:/app alpine
```

Show that the file is still there:

```
ls /app
```

Bind Mounts

Although you can, with a large amount of effort, get the data in a `docker volume` back out into your host machine, there is a much easier way. Instead of specifying a volume name, you can specify a folder on the host machine:

First create a file:

```
touch hostfile
```

Start a container mounting the current directory:

```
docker run -it -v ${PWD}:/app alpine
```

`${PWD}` will resolve to the absolute path of the current directory

Show that the file is now available in the container:

```
ls /app
```

Tmpfs Mounts

There's a third type of data volume you can use, but only on Linux. As the name would suggest this mount in transitory, it's tied to the lifecycle of the container. I.e. Once the container is stopped, the mount will no longer exist. As this mount is handled purely in memory it's used for storing sensitive information that you don't want to be persisted on either the host or container filesystems.