Chapter 6 - Persistence

Containers Are Designed To Be Ephemeral

Although we can start stopped containers, and pick up where we left off, that's an anti-pattern. Ideally we want repeatable operations on repeatable infrastructure.

Most worthwhile things are performing operations on data, and there are 3 ways to handle data volumes on containers.

Every time you run `docker run` it's a brand new container based on the `image`

The Container Lifecycle

Looking back at our `hello-world` container from earlier, **let's run it again:**

```
docker run --name myhelloworld hello-world
```

Now if we check the list of containers:

```
docker container ps
```
You'll notice it isn't there.

But if we get the list of all containers:

```
docker container ps -a
```

We can now see it:

```
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS                      PORTS               NAMES
f9850a139a45        hello-world         "/hello"                 32
seconds ago      Exited (0) 31 seconds ago
myhelloworld
```

We can see that it `Exited(0)`

This means the container is 'Stopped', stopped containers still exists on your disk, but are not using any other system resources. By default they will persist until you remove them.

Now compare that with our `alpine` containers:

```
docker run --name myalpine -dit alpine
```

Checking to see running containers:

```
docker container ps
```

We can see it's still running:

```
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
a71762793e40        alpine              "/bin/sh"           5 seconds ago
Up 4 seconds                            myalpine
```

What's the difference? Write down your ideas of why these two containers appear to act differently.

Naming Containers

In the last section we started containers specifying their names, `myhelloworld` and `myalpine`. What happens if we try to start a container with the same name?

```
docker run --name myalpine -dit alpine
```

We get an error.

Well it makes sense we can't have two running containers of the same name. So let's stop the running container:

```
docker container stop myalpine
```

Now if we try to run again:

```
docker run --name myalpine -dit alpine
```

We still get an error.

The rule is we can't overwrite a container, if you read the error message it says we need to remove the container to be able to reuse the name. So let's do that:

```
docker container rm myalpine
```

What scenarios can you think of where this might be a problem? Can you find in `docker run --help` a way of managing this?