# Chapter 9 - Networking

## Running Mynginx Image

Taking our `mynginx` image, the easiest way to access the server is to port-forward as we execute the `docker run` command.

**Start the container from the last chapter:**

```
docker run -d -p 80:80 mynginx
```

With the `-p` flag we have set so that port 80 on `localhost` is forward to port 80 on the container. Now we should be able to access `localhost` and see the file we put there earlier as part of the image build.

The `-d` flag says to run the container in `detached` mode, effectively in the background so we don't tie it to the current terminal session.

**Get the content from the server:**

```
curl localhost/index.html
```

## Container To Container Networking

If you have two containers that need to talk to each other, at first it can seem complicated. The containers run without knowing they're a container, and you can't address your way back to your host machine easily.

To show that it doesn't work:

**Start two alpine containers:**

```
docker run -dit --name alpine1 alpine
docker run -dit --name alpine2 alpine
```

**Attach to the first container:**

```
docker attach alpine1
```

**Ping the second container:**

```
ping alpine2
```

### Making It Work

Thankfully there's a construct called a docker network. This allows for simple intercontainer communication, by allowing addressing by container name.

To create a docker network, we can do it with a single command

**Create a docker network:**

```
docker network create mynetwork
```

**Start two alpine containers:**

```
docker run -dit --net mynetwork --name alpine3 alpine
docker run -dit --net mynetwork --name alpine4 alpine4
```

**Attach to the first container:**

```
docker attach alpine3
```

**Ping the second container:**

```
ping alpine4
```

You should now see responses from `alpine4`