# Chapter 8 - Dockerfiles

## Repeatable Images

Rather than building up images by hand, we can write `Dockerfiles`. These files are step by step instructions for building an image which are passed to the Docker Engine.

Let's create a quick html file to serve via Nginx:

> Create a file called mypage.html

> Add the following content

```
<html>
<body>
<p>Hello, Intro to Docker!</p>
</body>
</html>
```

> Create a new file called `Dockerfile`.

## The FROM Statement

The first line in a Dockerfile is the `FROM` statement. This specifies the image to build on top of.

> Add `FROM nginx:1.15.3` as the first line of your Dockerfile

## The COPY Statement

To add files to an image you can use the `COPY` statement.

> Append the line `COPY ./mypage.html /usr/share/nginx/html/mypage.html`

## The ADD Statement

A second way to add files is with the `ADD` statement.
`COPY` is preferred but add has one extra bit of functionality, you can specify a web address to download something from.

> e.g. `ADD https://www.docker.com /usr/share/nginx/html/docker.html`

## The RUN Statement

In order to run commands, such as installing packages, to a Docker image you use the `RUN` statement.

> Append the line `RUN apt-get update`

## The ENV Statement

If you need environment variables as part of an image you can specify them with the `ENV` statement

> Append the line `ENV USER <your name here>`

## The EXPOSE Statement

If you need to access the container via the network you need to specify what ports will be open. This is done with the `EXPOSE` statement.

> Append the line `EXPOSE 80`

## The ENTRYPOINT Statement

The `ENTRYPOINT` statement is used to specify fairly stable default commands for a container. For example the `maven` Docker images use:
`ENTRYPOINT ["/usr/local/bin/mvn-entrypoint.sh"]`
To execute the referenced script when the container starts.

> Append the line `ENTRYPOINT ["nginx"]`

## The CMD Statement

The `CMD` statement is used to specify the less stable default commands for a container. Often it is flags appended to the `ENTRYPOINT`

In our case we're going to pass a couple of options to our `nginx` call from our `ENTRYPOINT`

> Append the line `CMD ["-g", "daemon off;"]`

## Building An Image From A Dockerfile

To build an image we run the `docker build` command

> Run `docker build -t mynginx .`

In this example we have passed `-t mynginx` which names the image so we can reference it easier.

The `.` at the end of the line sets the current directory as what is known as the `build context` for the image.

### Build Contexts

To understand why build contexts are important, we need to understand how `docker build` functions.

- First it copies everything in the build context across to a folder in `/tmp`
- Then it performs the steps in the Dockerfile, performing a `docker commit` after every line
- Then it names and tags the image, making it available for use

Therefore, the build context are the files which are copied over to the temporary folder, and define what files are accessible for the statements in the `Dockerfile`

## Running This Image

In order to run this image, we need to talk about networking, which is the topic of the next chapter.