In this scenario, you'll learn how to create a Docker Image for running a static HTML website using Nginx. The scenario will explain how to build a Docker Image running Nginx with your HTML site.

The aim is to help you understand how to create and run Docker Images created by yourself.

Step 1 - Create Dockerfile
Docker Images start from a base image. The base image should include the platform dependencies required by your application, for example, having the JVM or CLR installed.

This base image is defined as an instruction in the Dockerfile. Docker Images are built based on the contents of a Dockerfile. The Dockerfile is a list of instructions describing how to deploy your application.

In this example, our base image is the Alpine version of Nginx. This provides the configured web server on the Linux Alpine distribution.

Task
Create your Dockerfile for building your image by copying the contents below into the editor.

FROM nginx:alpine
COPY . /usr/share/nginx/html

The first line defines our base image. The second line copies the content of the current directory into a particular location inside the container.

Step 2 - Build Docker Image
The Dockerfile is used by the Docker CLI build command. The build command executes each instruction within the Dockerfile. The result is a built Docker Image that can be launched and run your configured app.

The build command takes in some different parameters. The format is docker build -t <build-directory>. The -t parameter allows you to specify a friendly name for the image and a tag, commonly used as a version number. This allows you to track built images and be confident about which version is being started.

Task
Build our static HTML image using the build command below.

docker build -t webserver-image:v1 .

You can view a list of all the images on the host using - docker images.

The built image will have the name webserver-image with a tag of v1.

```
 => => naming to docker.io/library/webserver-image:v1
root@host01:~$ docker images
REPOSITORY        TAG       IMAGE ID       CREATED          SIZE
webserver-image   v1        6eab4a6d0d8c   14 seconds ago   41MB
root@host01:~$
```

Step 3 - Run
The built Image can be launched in a consistent way to other Docker Images. When a container launches, it's sandboxed from other processes and networks on the host. When starting a container you need to give it permission and access to what it requires.

For example, to open and bind to a network port on the host you need to provide the parameter -p <host-port>:<container-port>.

Task
Launch our newly built image providing the friendly name and tag. As it's a web server, bind port 80 to our host using the -p parameter.

docker run -d -p 80:80 webserver-image:v1

Once started, you'll be able to access the results of port 80 via - curl host01

```
webserver-image   v1          6eab4a6d0d8c   14 seconds ago   41MB
root@host01:~$ docker run -d -p 80:80 webserver-image:v1
ec71b4a796b0dc6ee32edd0db2bb600463b3e6aec00b2b3a87e4d3379732e852
root@host01:~$ curl host01
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
```

To render the requests in the browser use the following links

https://5af8c269895a4a08b0bc22bd25db3955-2887242757-80-host09nc.environments.katacoda.com/

You now have a static HTML website being served by Nginx.