# Getting Started with Docker

## Overview

In this lab, you'll download the nginx image from Docker Hub. nginx is a free, lightweight, open-source web server and is widely used in the industry. You'll run several containers based on this image, and explore how to interact with these containers.

## Roadmap

There are 4 exercises in this lab, of which the last exercise is "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow in the rest of this document:

1. Pulling an nginx image from Docker Hub
2. Running nginx containers
3. Investigating containers
4. (If Time Permits) Interacting with a running nginx container

## Exercise 1: Pulling an nginx image from Docker Hub

Open a browser and go to Docker Hub at <https://hub.docker.com>, and search for nginx. View all of the available tags (i.e. variants) of nginx. From a terminology point of view:

* nginx is a repository.
* The nginx repository contains lots of images. Each image has a separate tag name. For example, the nginx repository has images with tags such as stable-alpine-perl, stable-alpine, etc.

When you pull an image from Docker Hub (or any other registry), you specify a repository name and optionally a tag. For example, in your Linux box, run the following command; the command pulls the nginx image tagged with the name stable-alpine-perl:

## docker pull nginx:stable-alpine-perl

If you omit the tag, Docker downloads the image tagged with the name latest by default. To see this, run the following command:

## docker pull nginx

## You can list all the images in your local Docker registry as follows:

docker image ls

If you want to list the images for a particular repository (e.g. all the images from the nginx repository, each with a different tag), just append the repository name in the ls command. For example, run the following command to lists all images for the nginx repository (i.e. all the different versions of nginx images) in the local Docker registry:

docker image ls nginx

**Exercise 2: Running nginx containers**

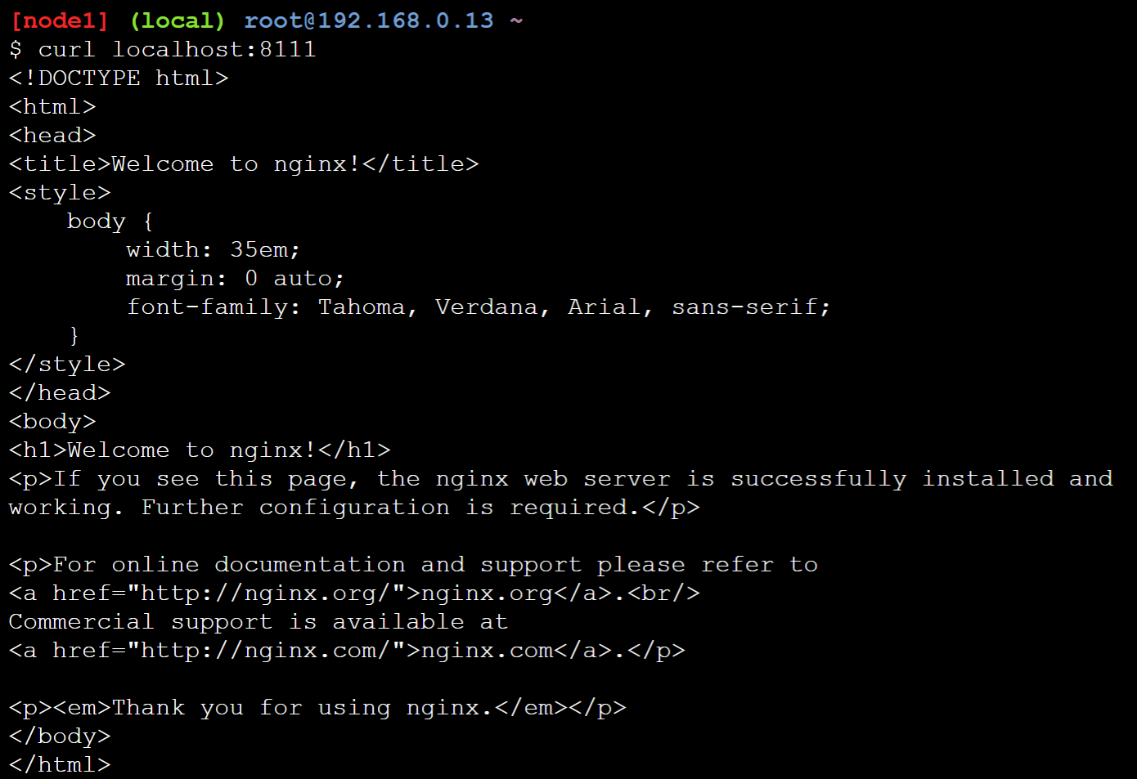
nginx listens on port 80. If you want to run it in a Docker container, you must map this port to an available port on your host computer. For example, run the following command to start an nginx container that maps host port 8111 to nginx port 80:

docker run --name mynginx1 -d -p 8111:80 nginx

To verify nginx is running in the container, run the following command:

curl localhost:8111

This sends an HTTP request to port 8111 on localhost (i.e. your Linux machine). This is mapped to port 80 in the container, which is the port nginx is listening on. You should see the HTML for the nginx home page, courtesy of the nginx web server listening on port 80 in the container.



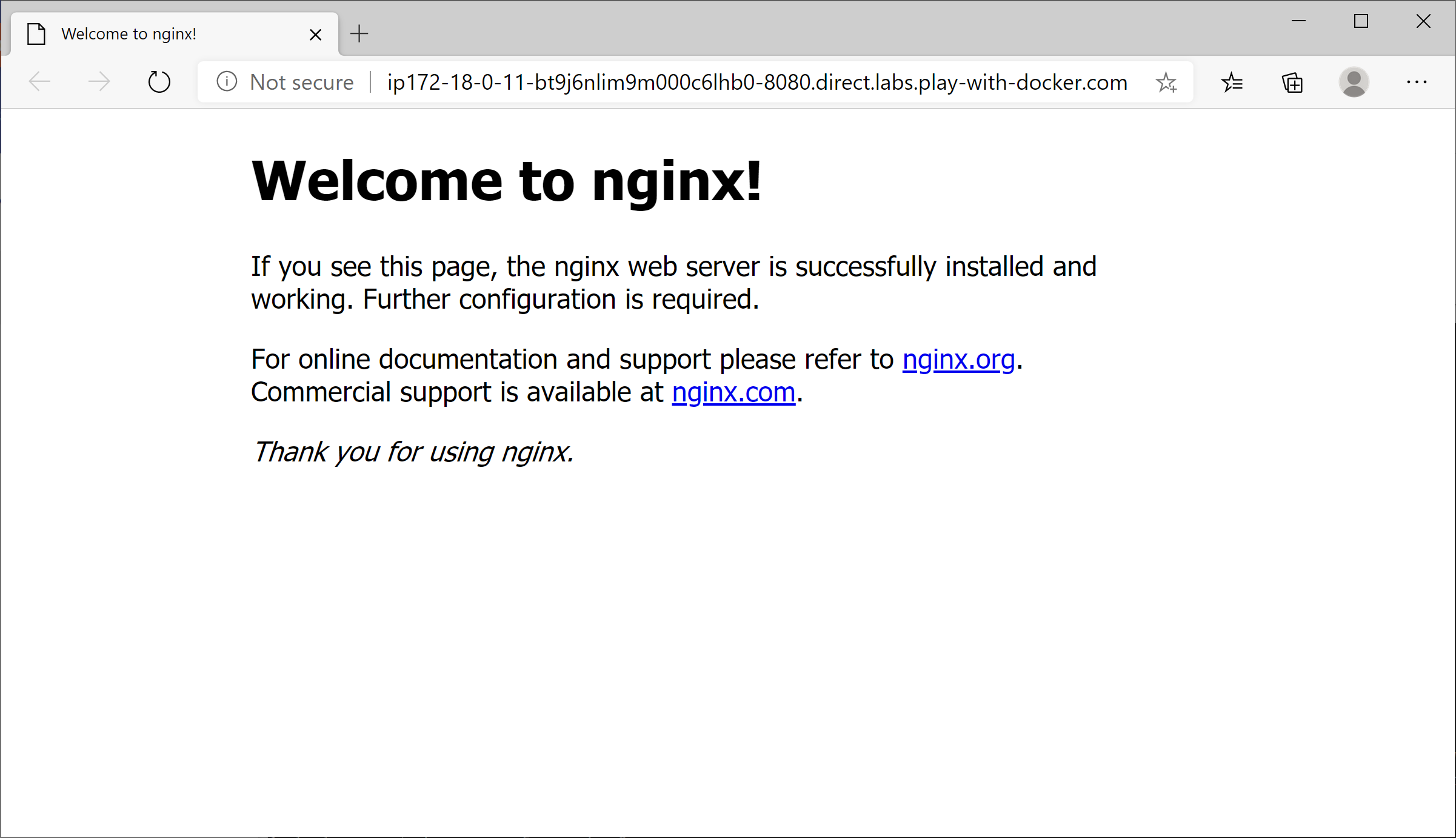
Now start another nginx container that maps host port 8080 to nginx port 80:

docker run --name mynginx2 -d -p 8080:80 nginx

Note: If you get an error indicating port 8080 is already bound, you probably have a container running Tomcat from the chapter, so remove it via *docker container rm -f containerName*.

Use curl to ping localhost:8080; you should see the nginx home page, courtesy of the nginx web server listening on port 80 in this new container.

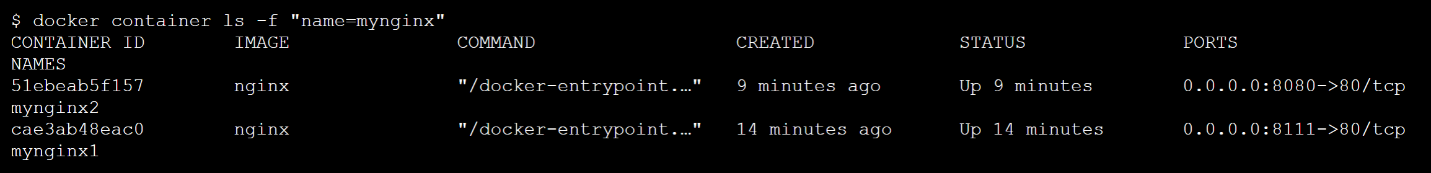
Now try to access the nginx via a web browser; see the chapter for a reminder about how to open a browser on port 8080 in the Play for Docker environment. The browser should display the nginx home page as follows:



## Exercise 3: Investigating containers

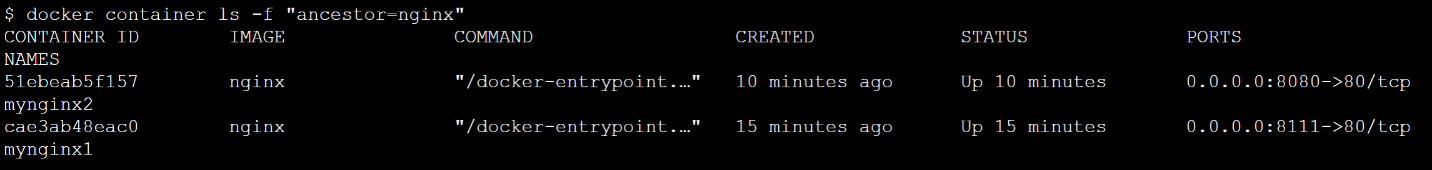
You currently have two containers running. Both containers are instances of the nginx image (or more correctly, both are instances of the nginx:latest image). The containers are named mynginx1 and mynginx2. If you want to list just these containers, you can specify a filter via the --filter flag (or –f for short). For example:

docker container ls -f "name=mynginx"



You can also filter containers based on the *image* name, via the ancestor property. This kinda makes sense (the image is kinda the ancestor of the running instances ☺):

docker container ls -f "ancestor=nginx"

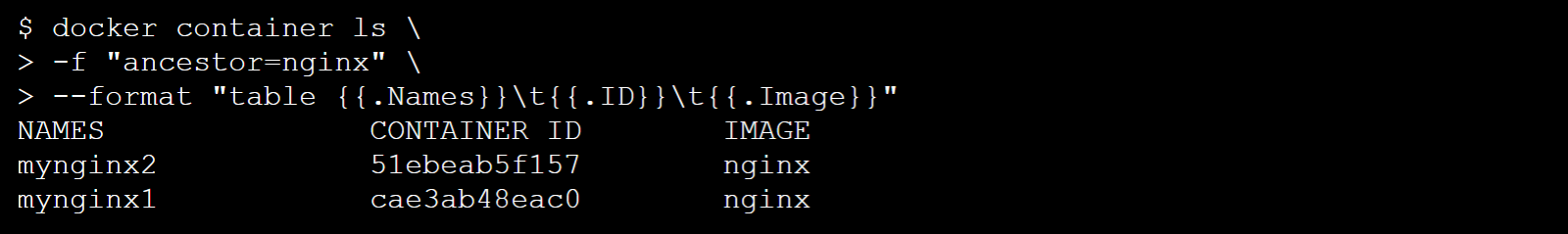


You can also format the output via the --format flag, to just show the information you're interested in. For example, to show the container name(s), container id, and image name, run the following command (note the \ at the end of each line except the last line):

docker container ls **\**

-f "ancestor=nginx" **\**

--format "table {{.Names}}\t{{.ID}}\t{{.Image}}"



## Exercise 4 (If time permits): Interacting with a running nginx container

In this exercise you will execute interactive commands in a container. To facilitate this, run the following command to open an interactive shell in the mynginx1 container:

docker container exec -it mynginx2 /bin/sh

You should see a # shell prompt. Congratulations, you are now inside the container! Run the following commands, one after the other, to probe the directory structure within the container:

ls

ls /usr

ls /usr/share

ls /usr/share/nginx

ls /usr/share/nginx/html

cat /usr/share/nginx/html/index.html

What the previous commands show is that the container has a file named index.html in the directory /usr/share/nginx/html. This is the home page for the nginx web server. You can create new web pages in this directory too. For example, run the following commands to create a super-simple HTML file named greeting.html:

cd /usr/share/nginx/html

echo "<html><body>Bore da pawb</body></html>" >greeting.html

exit

Now use curl to ping the following URL, to can see the fruit of your work:

curl localhost:8080/greeting.html

Also browse to greeting.html in a browser, via the IP address for your Linux box.

