Module 8 – Networking: Lab

# Lab Objectives

Accessing a deployed application on the host

Configuring an NGINX reverse proxy for an application

1. Accessing a deployed application

1. Create a new folder (~/docker/networking\_exercise) for this exercise and make sure that you are running your commands and creating the required files in this folder.
2. We’ll deploy the default NGINX image and attach the container to the host network. Once the container has started you will be able to access it in your browser by using the following command:

**docker run -d --network host --name nginx nginx**

1. Stop and remove the nginx container when you are done.
2. We’ll create a new bridged network for our reverse proxy setup by using the following command:

**docker network create my-network**

1. For this example we can use Jenkins which is a very popular CI tool that is web based, you may deploy your own application if you wish though. Create the application container and attach it to the new bridge network by using the following command:

**docker run -d --network my-network --name jenkins Jenkins**

2. Creating an NGINX application

1. NGINX is another very popular tool that can address many networking issues, we’ll be using it for it’s primary purpose here, as a reverse proxy. We’ll need create our own config file for this (~/docker/networking\_exercise/nginx.conf). The config file should contain the following lines:

**events {}**

**http {**

**server {**

**listen 80;**

**location / {**

**proxy\_pass http://jenkins:8080;**

**}**

**}**

**}**

1. We are going to pass in the configuration that we made as volume to the container, this handout doesn’t explain what this is however all you need to know at this point is that it gives the container access to the configuration file that we just created. The NGINX container that we create must also be attached to the network we made earlier We’ll also publish port 80 we can access NGINX from outside of the bridge network.

**docker run -d --network my-network -v**

**$(pwd)/nginx.conf:/etc/nginx/nginx.conf -p 80:80 --name nginx nginx**

1. You should now be able to access your application from the host, <http://localhost>, or if it a remote server with port 80 opened on the firewall you can just access your application via the public IP address.
2. When you connect to your application the traffic is going through NGINX first, it is then routed to you deployed application even though they are in separate containers. This made possible because both the containers are on the same bridge network.

3. Finish up

1. Stop and remove all the containers, delete the bridge network that was created.
2. Can you get this working on one of your own projects? For instance a server that connects to a database, or a frontend application that connects to a server using Docker networks.