**TRAINING MATERIALS - MODULE HANDOUT**

**Contacts**

**robert.crutchley@qa.com**

*team.qac.all.trainers@qa.com*

*www.consulting.qa.com*

**Contents**

[**Overview**](#_yrn0ksxn3au) **1**

[**Networks in Docker**](#_iypcftccydbf) **2**

[Bridged Networks](#_s75ub7knhqjg) 2

[Host Networks](#_7rhupy1m9reo) 2

[Overlay Networks](#_tdz75tqr4uwx) 2

[Macvlan Networks](#_wsp0qjj85j7n) 2

[**Managing Docker Networks**](#_2ffoutg0c6q7) **2**

[Connecting a New Container to a Network](#_2s05hvlrp2xg) 2

[Connect an Already Existing Container to a Network](#_b03pqgrvqan4) 2

[Disconnect a Container from a Network](#_pis4x64zx1fi) 3

[Delete a Network](#_r1ripoam80zj) 3

[Viewing your Docker Networks](#_tu7utv29falx) 3

[**Bridged Networks**](#_jkpbec5bug3z) **3**

[Overview](#_tefky72806yf) 3

[Create Bridged Networks](#_h2cx1xyzqt1l) 4

[**Host Networks**](#_pnbkfbrxkq9e) **4**

[**Tasks**](#_891drw7fw37s) **4**

[Creating a Container on the Host Network Driver](#_mjm0sg76w5ob) 4

[Create a New Bridged Network](#_76o2k26tnjcl) 4

[Create an Application Container](#_q5wa69momtb3) 5

[Create an NGINX Configuration](#_1azcaevxcf86) 5

[Create an NGINX Container Using Our Configurations](#_96fp64sb1qe9) 5

[Access Your Application](#_b376iubnz273) 5

[Cleanup](#_8y39vmtclbkk) 6

# Overview

We can utilize networking in Docker when we would like to have multiple containers working together. For example, a frontend and backend application with a database. The frontend application needs to be able to communicate with the backend and the backend application needs to be able to communicate with the database.

We can have these three different applications running in separate containers. This is because if they were all in the same container, then everything would need to be redeployed whenever any of the three have been updated.

# Networks in Docker

There are several different types of network drivers, for different networking purposes. Some are for single host solutions while others are for spreading containers across more than one host. The main focus for us will be with bridge networks because overlay networks are for Docker Swarm setups and macvlan networks are for legacy applications.

## Bridged Networks

Bridged networks are for connecting multiple containers on a single host. You will be able to access the container with it’s private IP address. Mapping ports to the host needs to be done explicitly.

## Host Networks

These should be used when you want container application ports to map to the host automatically. Instead of explicitly publishing a port from a container to the host, applications that listen on a port will be available from the host on that same port. Be care of ports clashing between applications when you use this.

## Overlay Networks

This is where Docker allows you to scale your application, overlay networks allows network connectivity over multiple hosts. Your applications deployed on different hosts will be able to communicate to each other, this kind of networking appears when using Docker Swarm, a container orchestration tool but can also be used for standalone containers.

## Macvlan Networks

Some legacy applications or tools which monitor network traffic expect to be on a physical network, which Docker networks are definitely not. Macvlan networks are for these kinds of applications.

# Managing Docker Networks

## Connecting a New Container to a Network

Containers can set to connect to a network when you are creating them by providing the **--network** flag.

|  |
| --- |
| docker network create **[NETWORK\_NAME]** |
| docker network create my-network |

## Connect an Already Existing Container to a Network

If you have a container that exists already but you want to add it to a network without having to recreate it then you can use the **network connect** command, providing the network name and the container name.

|  |
| --- |
| docker network connect **[NETWORK\_NAME] [CONTAINER\_NAME]** |
| docker network connect my-network my-container |

## Disconnect a Container from a Network

Disconnecting a container from a network can be done just like when you connect an existing container to a network, provide the network name and the container name.

|  |
| --- |
| docker network disconnect **[NETWORK\_NAME] [CONTAINER\_NAME]** |
| docker network disconnect my-network my-container |

## Delete a Network

You can only delete a network if there are not containers connected to it otherwise you will get an error message.

|  |
| --- |
| docker network rm **[NETWORK\_NAME]** |
| docker network rm my-network |

## Viewing your Docker Networks

You can view what networks you already have along with their details such as what type of network they are and the name of them.

|  |
| --- |
| docker network list |

# Bridged Networks

## Overview

There are two kinds of bridged networks, user defined and default. When containers are started a **--link** flag can be provided to link it to another container using the default bridge network. The default bridge is legacy method in Docker for networking and is not recommended anymore by Docker, so we shall avoid that going forward.

User defined bridged networks provide many benefits

* **Isolation from other containers on the same host that aren’t in the same network.**  
  When two containers are on the same user defined bridge network, all ports are open to each other and they can communicate freely.
* **Automatic DNS Resolution**  
  When a container is put on a bridge network it will have a private IP address. A DNS entry is automatically configured to this IP address which is just the container name. For instance if there were two containers running, one called **nginx** and the other called **application**, and the application container had a server listening on port 8080. The nginx container would be able to connect to the application running in the other container by its name ([**http://application:8080**](http://application:8080)).
* **Connect and Disconnect Containers on the Fly**  
  You can connect and disconnect containers to a bridge network without having to stop the containers.

## Create Bridged Networks

When you create a network with the **docker network** command the default network type is bridged. To create a bridged network you need to provide the name of the new network.

|  |
| --- |
| docker network create **[NETWORK\_NAME]** |
| docker network create my-network |

# Host Networks

Something to keep in mind with host networks is at the time of writing this host networks are only available on Linux hosts running Docker. If you list the networks that you have you should see that there is one network called host. To get a container onto this network you can just specify the network name when connecting the container to it.

to a server using Docker networks.