This scenario explores how to create a docker network allowing containers to communicate. We'll also explore the Embedded DNS Server added in Docker 1.10.

Docker has two approaches to networking. The first defines a link between two containers. This link updates /etc/hosts and environment variables to allow containers to discover and communicate.

The alternate approach is to create a docker network that containers are connected to. The network has similar attributes to a physical network, allowing containers to come and go more freely than when using links.

Docker Networks

Step 1 - Create Network

The first step is to create a network using the CLI. This network will allow us to attach multiple containers which will be able to discover each other.

In this example, we're going to start by creating a backend-network. All containers attached to our backend will be on this network.

Task: Create Network

To start with we create the network with our predefined name

docker network create backend-network

Task: Connect To Networks

When we launch new containers, we can use the --net attribute to assign which network they should be connected to.

docker network ls

docker run -d --name=redis --net=backend-network redis

docker network ls

Step 2 - Network Communication

Task: Explore

docker run --net=backend-network alpine env

docker run --net=backend-network alpine cat /etc/hosts

docker run --net=backend-network alpine cat /etc/resolv.conf

docker run --net=backend-network alpine ping -c1 redis

Step 3 - Connect Two Containers

Task

The first task is to create a new network in the same way.

docker network create frontend-network

docker network connect frontend-network redis

docker run -d -p 3000:3000 --net=frontend-network katacoda/redis-node-docker-example

curl docker:3000

Step 4 - Create Aliases

Connect Container with Alias

The following command will connect our Redis instance to the frontend-network with the alias of db.

docker network create frontend-network2

docker network connect --alias db frontend-network2 redis

When containers attempt to access a service via the name db, they will be given the IP address of our Redis container.

docker run --net=frontend-network2 alpine ping -c1 db

Step 5 - Disconnect Containers

The following command will list all the networks on our host.

docker network ls

We can then explore the network to see which containers are attached and their IP addresses.

docker network inspect frontend-network

The following command disconnects the redis container from the frontend-network.

docker network disconnect frontend-network redis