

**Lab Manual- Docker Network**

**Prepared for**:

**Date:** 18th Dec 2023

**Prepared by:**

Document Name: Lab Manual **Document Number** AZLabn912

**Contributor:**

Contents

[1. Objective 3](#_Toc155432767)

[2. Use the default bridge network 3](#_Toc155432768)

[3. Use the User-defined bridge network 6](#_Toc155432769)

# Objective

This Lab deals with networking for standalone Docker containers This topic includes two different lab

**Use the default bridge network** demonstrates how to use the default bridge network that Docker sets up for you automatically. This network is not the best choice for production systems.

**Use user-defined bridge networks** shows how to create and use your own custom bridge networks, to connect containers running on the same Docker host. This is recommended for standalone containers running in production.

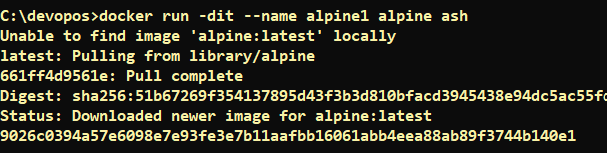
Although overlay networks are generally used for swarm services, you can also use an overlay network for standalone containers.

# Use the default bridge network

In this example, you start two different alpine containers on the same Docker host and do some tests to understand how they communicate with each other. You need to have Docker installed and running.

* Start two **alpine containers** running **ash**, which is Alpine's default shell rather than bash. The -dit flags mean to start the container detached (in the background), interactive (with the ability to type into it), and with a TTY (so you can see the input and output)

docker run -dit --name alpine1 alpine ash

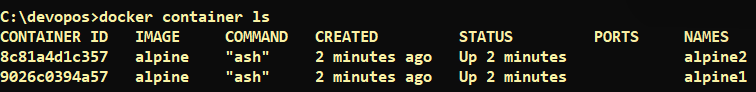


docker run -dit --name alpine2 alpine ash



* Check that both containers are actually started:

docker container ls



* Inspect the bridge network to see what containers are connected to it.

docker network inspect bridge



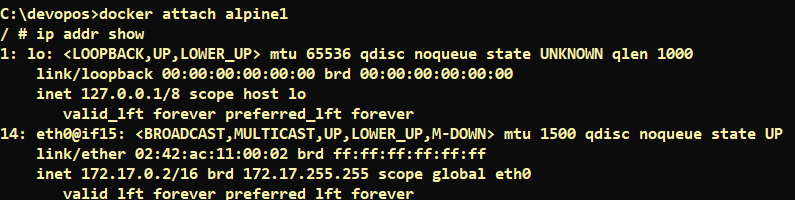
**Note :** Near the top, information about the bridge network is listed, including the IP address of the gateway between the Docker host and the bridge network (172.17.0.1). Under the Containers key, each connected container is listed, along with information about its IP address (172.17.0.2 for alpine1 and 172.17.0.3 for alpine2).

* The containers are running in the background. Use the **docker attach** command to connect to **alpine1**.

docker attach alpine1

The prompt changes to # to indicate that you are the root user within the container. Use the ip addr show command to show the network interfaces for alpine1 as they look from within the container:

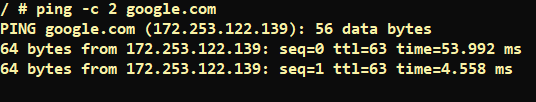
ip addr show



**Note :** The first interface is the **loopback device**. Ignore it for now. Notice that the **second interface** has the IP address 172.17.0.2, which is the same address shown for **alpine1** in the previous step.

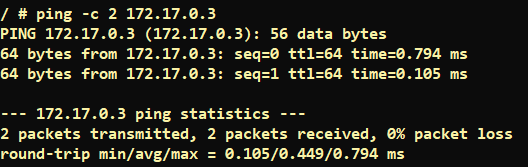
* From within alpine1, make sure you can connect to the internet by pinging google.com. The -c 2 flag limits the command to two ping attempts

ping -c 2 google.com



* Now try to ping the second container. First, ping it by its IP address, 172.17.0.3:

ping -c 2 172.17.0.3



* This succeeds. Next, try pinging the alpine2 container by container name. This will fail.

**ping -c 2 alpine2**

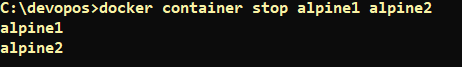
* Now exit

exit



* Stop and remove both containers.

docker container stop alpine1 alpine2



docker container rm alpine1 alpine2



# Use the User-defined bridge network

In this example, we again start two alpine containers, but attach them to a user-defined network called alpine-net which we have already created. These containers are not connected to the default bridge network at all. We then start a third alpine container which is connected to the bridge network but not connected to alpine-net, and a fourth alpine container which is connected to both networks.

* Create the **alpine-net** network. You do not need the **--driver bridge** flag since it's the default, but this example shows how to specify it.

docker network create --driver bridge alpine-net

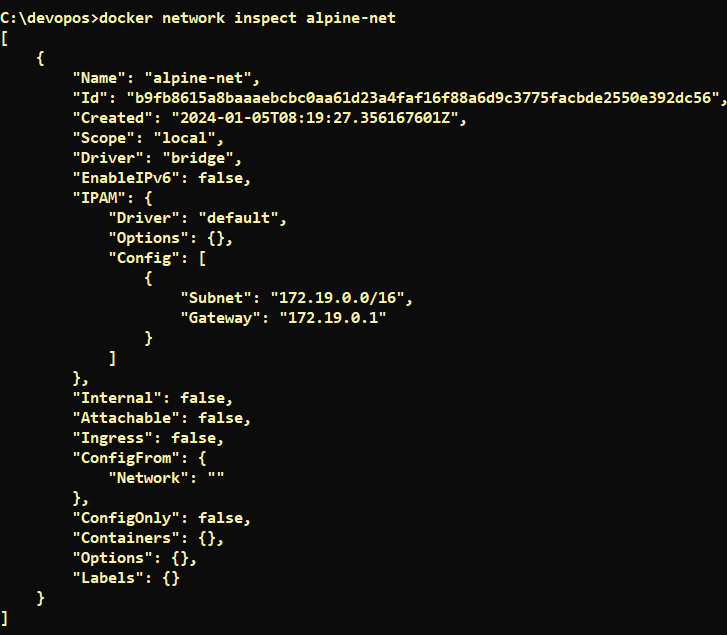


* List Docker's networks:

docker network ls

* Inspect the **alpine-net** network. This shows you its IP address and the fact that no containers are connected to it

docker network inspect alpine-net



**Note:** Notice that this network's gateway is 172.18.0.1, as opposed to the default bridge network, whose gateway is 172.17.0.1. The exact IP address may be different on your system.

* Create your **four containers**. Notice the **--network** flags. You can only connect to one network during the docker run command, so you need to use **docker network connect** afterward to connect alpine4 to the **bridge network** as well.

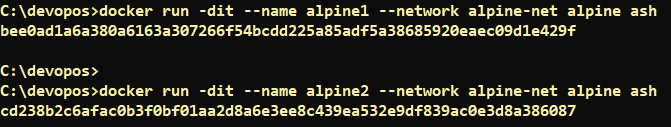
docker run -dit --name alpine1 --network alpine-net alpine ash

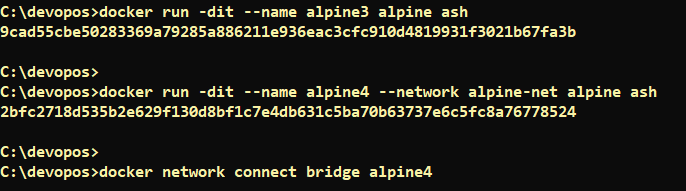
docker run -dit --name alpine2 --network alpine-net alpine ash

docker run -dit --name alpine3 alpine ash

docker run -dit --name alpine4 --network alpine-net alpine ash

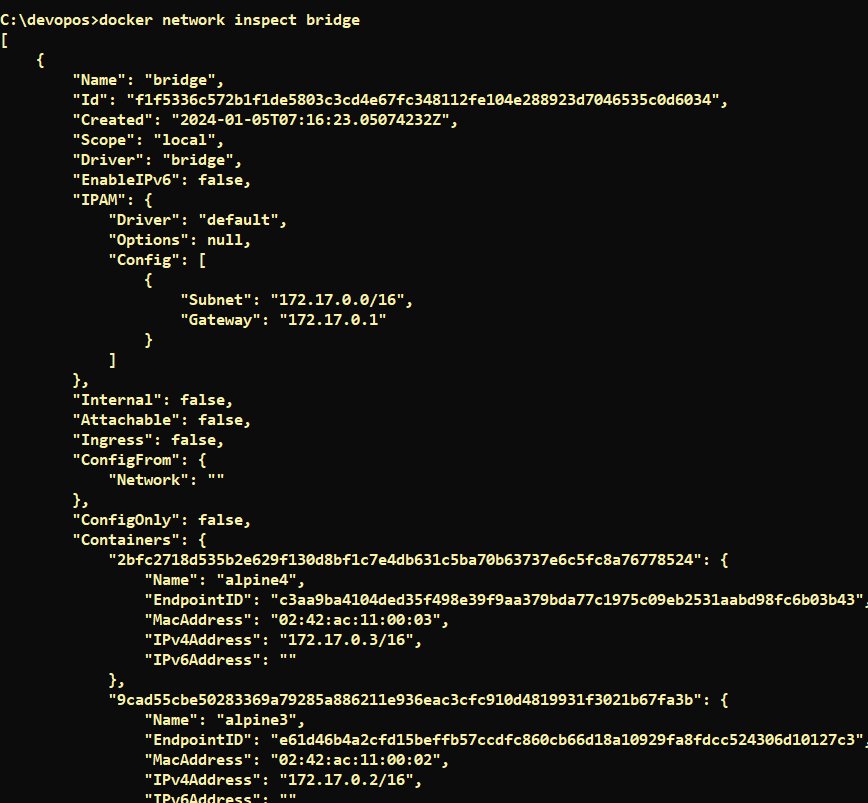
docker network connect bridge alpine4





* Inspect the **bridge network** and the alpine-net network again

docker network inspect bridge



Note : Containers alpine3 and alpine4 are connected to the bridge network.

* Notw Inspect alpine-net

docker network inspect alpine-net



Containers alpine1, alpine2, and alpine4 are connected to the alpine-net network.

* On user-defined networks like alpine-net, containers can not only communicate by IP address, but can also resolve a container name to an IP address. This capability is called **automatic service discovery**. Let's connect to alpine1 and test this out. alpine1 should be able to resolve alpine2 and alpine4 (and alpine1, itself) to IP addresses.

docker container attach alpine1

