# Learning Objective

Enable inter-container communication.

# Introduction

This lab provides you with an example to enable inter-container communication in Docker.

We shall re-use the Book service which was deployed earlier and also deploy the python program\* test\_invoke\_http.py (and invokes.py) which you have used in an earlier lab to invoke (communicate with) the Book service.

*\* We shall name this image callbook*

# Preparation

1. Create a project folder named **callbook** in C:\ESD\Labs\docker:

**mkdir C:\ESD\Labs\docker\callbook**

1. Obtain the following files from Google drive and place the contents into the new folder:

* invokes.py
* test\_invoke\_http.py
* Dockerfile
* requirements.txt

1. Open the project folder in VS Code and inspect the files:

**cd C:\ESD\Labs\docker\callbook**

**code .**

*Dockerfile and requirements.txt should look familiar to you; you have used similar ones when deploying the Book service with the following changes:*

* ***Dockerfile****, we are copying 2 files into the image:*

*COPY ./invokes.py ./test\_invoke\_http.py ./*

* ***invokes.py*** *uses the requests library to invoke the Book service and return the responses. Hence, the following was added to requirements.txt:*

requests==2.28.2

* *In* ***test\_invoke\_http.py****, to make it easier for testing, we shall also use the input () function in python as an alternative to setting the environment variable for the Book service URL:*

*bookURL = environ.get('bookURL') or input("Enter Book service URL: ")*

* + *If the environment variable is set, the program will use its value. Otherwise, it will prompt for an input.*

# Create Docker image

1. In VS Code, create a terminal and change directory to the project folder:

**Terminal → New Terminal**

**cd C:\ESD\Labs\docker\callbook**

1. Create the image by entering:

*(replace the <dockerid> with yours)*

docker build -t <dockerid>/callbook:1.0 **.**/

# 

# Configure Networking and Test

1. If you have not deleted the Book service containers created previously, delete the containers by entering the following commands in VS Code terminal:

docker ps -a

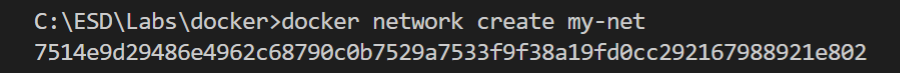
*(-a to list all the containers; then)*

docker rm -f <containerid1> <containerid2>

*rm -f will remove the containers even if they are still running. So there is no need to stop them first*

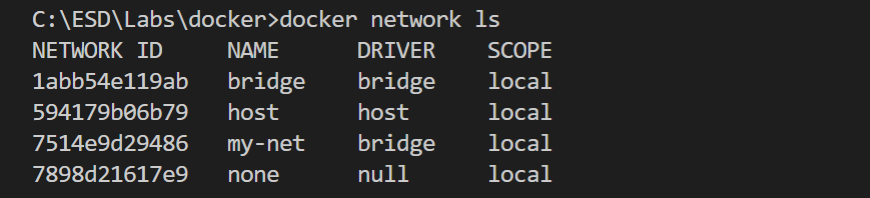
1. Create a custom Docker network named “my-net” to allow the containers to discover and communicate with each other:

docker network create my-net



*You can list the available networks by entering the following command:*

docker network ls

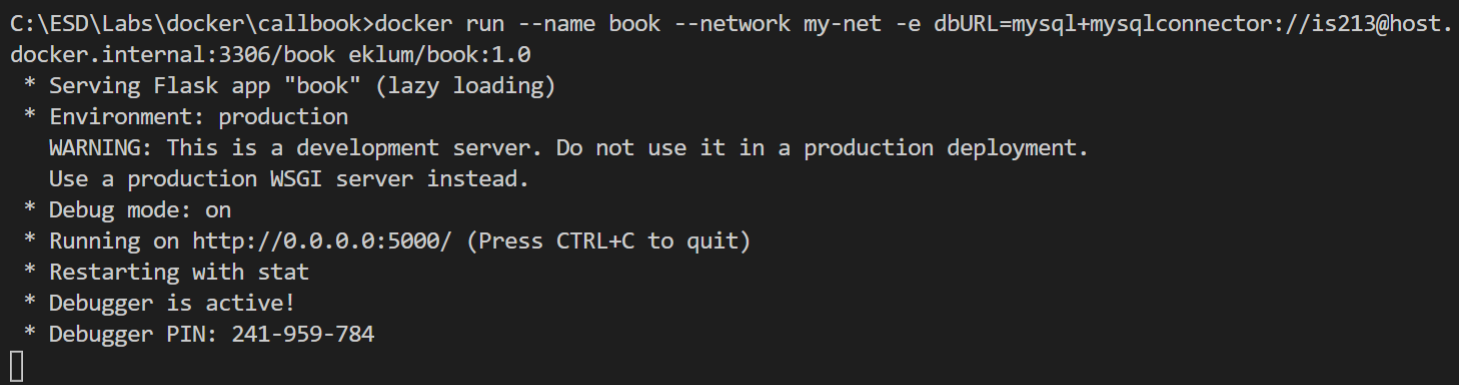
**

*You should see the new network in the list. Unless otherwise specified in the command, the default bridge network (also named “bridge”) is used.* (Note: Containers on the default bridge network can only access each other by IP addresses, while on a custom bridge network, containers can resolve each other by name or alias; see [here](https://docs.docker.com/network/bridge/).)

*Bridge networks are usually used when your applications run in standalone containers that need to communicate. For more information, read* [*here*](https://docs.docker.com/network/)*.*

1. Then, run a Book service container by entering the following command in one line *(replace the <dockerid> with yours):*

docker run **--name book --network my-net** -e dbURL=mysql+mysqlconnector://is213@host.docker.internal:3306/book <dockerid>/book:1.0



* *We name the container as book (Docker will generate a random name if the --name option is not specified)*
* *We connect the container to the my-net network*

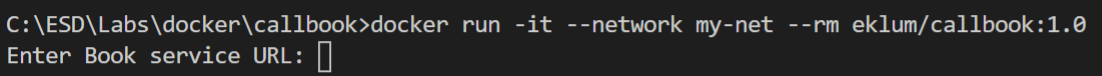
*Contrast this with the command you have used previously:*

*docker run* ***-p 5000:5000*** *-e dbURL=mysql+mysqlconnector://is213@host.docker.internal:3306/book <dockerid>/book:1.0*

* *We no longer publish the container’s port to the host by removing the -p option. If you attempt to access the book service via* [*http://localhost:5000/book*](http://localhost:5000/book)*, it will no longer work.*

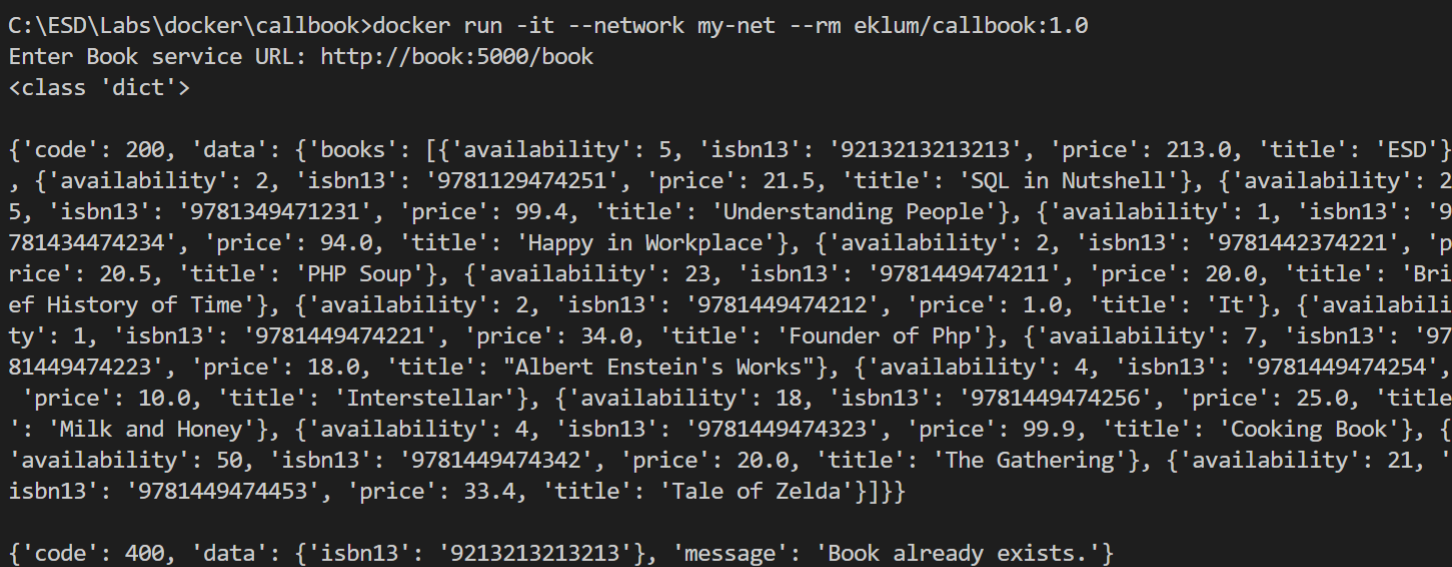
1. Now, in another VS Code terminal window, run the callbook container by entering the following command in one line *(replace the <dockerid> with yours):*

docker run **-it --network my-net** **--rm** <dockerid>/callbook:1.0

**

* *We use the -it option to run the container in interactive mode so that we can enter the Book service URL*
* *We connect the container to the my-net network*

Type **http://book:5000/book** for the Book service URL and press Enter:



*You should see responses from the Book service if networking is configured correctly. If you get a 400 error code for adding a new book (when the book already exists), you can ignore it.*

* *We use* ***book*** *(instead of IP address or localhost) as the hostname of the Book service (i.e. http://****book****:5000/book); this works because of the --name option when running the book container*
* *We use the --rm option to make Docker automatically remove the container when it exits*

*If you now list the containers using docker ps -a, you would see no trace of the callbook container; this is convenient for automatically removing temporary containers.*

1. When you are done, remove the Book service container and the network:

docker rm -f book

docker network rm my-net

# Learning Points

* *Enable inter-container communication via a custom network and container names*
* *Create, delete custom network*
* *Work with Docker commands and options (docker run (--name, --network, -it, --rm options))*