# Learning Objective

Run multi-container Docker applications with [Compose](https://docs.docker.com/compose/).

Compose is a tool that uses a YAML file to configure your application’s services. Then, with a single command, you create and start all the services from your configuration.

# Introduction

This lab provides you with an example to create and start multiple containers with Compose.

# Create the Compose file

1. In VS Code, open the project folder C:\ESD\Labs\docker
2. Create a new file in your project folder named **docker-compose.yml**
3. In docker-compose.yml, paste the following:

version: "3.8"

services:

#################################

# Book: The Book microservice

#################################

book:

image: <dockerid>/book:1.0

restart: always

environment:

dbURL: mysql+mysqlconnector://is213@host.docker.internal:3306/book

###############################################

# callbook: The test\_invoke\_http.py program

###############################################

callbook:

image: <dockerid>/callbook:1.0

depends\_on:

- book

environment:

bookURL: http://book:5000/book

*Replace the <dockerid> with yours and change the dbURL if your settings are different. E.g. port number*

Explanation

*Ref:* [*https://docs.docker.com/compose/compose-file/compose-file-v3/*](https://docs.docker.com/compose/compose-file/compose-file-v3/)

version: "3.8"

*The version of the compose file format to use. This is dependent on the Docker engine’s version you are using.*

services:

*This is the section where we define our services* ***book*** *and* ***callbook****. You can think of a service (in the compose context) as a container.*

#################################

# Book: The Book microservice

#################################

book:

image: <dockerid>/book:1.0

restart: always

environment:

dbURL: mysql+mysqlconnector://is213@host.docker.internal:3306/book

* *We define a service named* ***book***
* *We specify the image to start the container from - <dockerid>/book:1.0*
* *We want to always (auto) restart if the container stops*
* *We set the environment variable dbURL with the value specified*

###############################################

# callbook: The test\_invoke\_http.py program

###############################################

callbook:

image: <dockerid>/callbook:1.0

depends\_on:

- book

environment:

bookURL: http://book:5000/book

* *We define a service named* ***callbook***
* *We specify the image to start the container from - <dockerid>/callbook:1.0*
* *We use depends\_on to express dependency between services so as to control the startup and shutdown order. In this case, we want to start the book service before starting this service.*
* *Note that for startup, Compose does not wait until a container is “ready” - only until it’s running. For these 2 services, that is fine because the Book service container starts up relatively quickly.*

*If you need to wait for a service to be ready, see* [*Control startup order*](https://docs.docker.com/compose/startup-order/) *for more on this problem and strategies for solving it.*

* *We set the environment variable bookURL with the value specified.*

1. Save the .yml file

# Run application with Compose

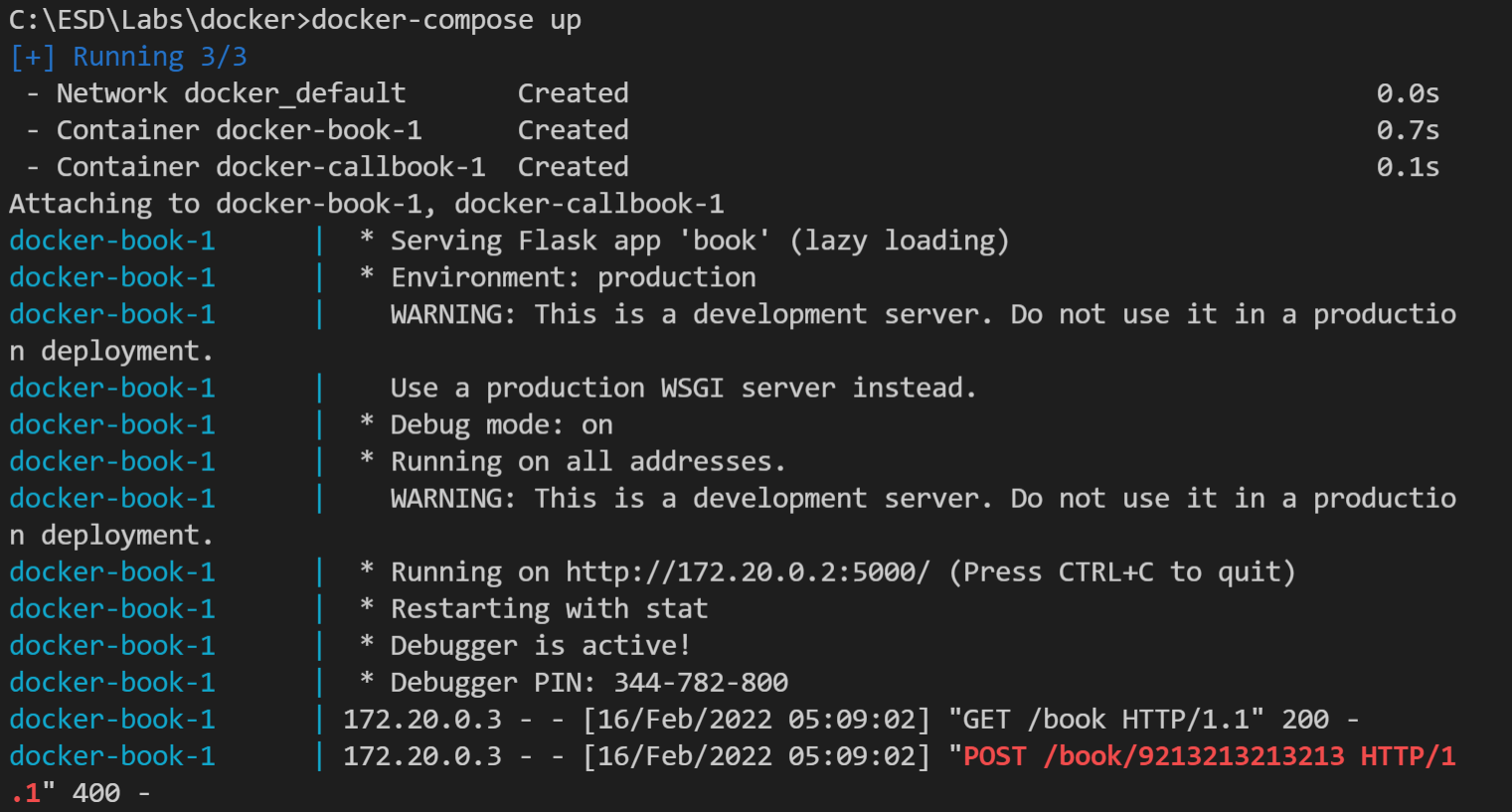
1. In VS Code, create a terminal and change directory to the project folder:

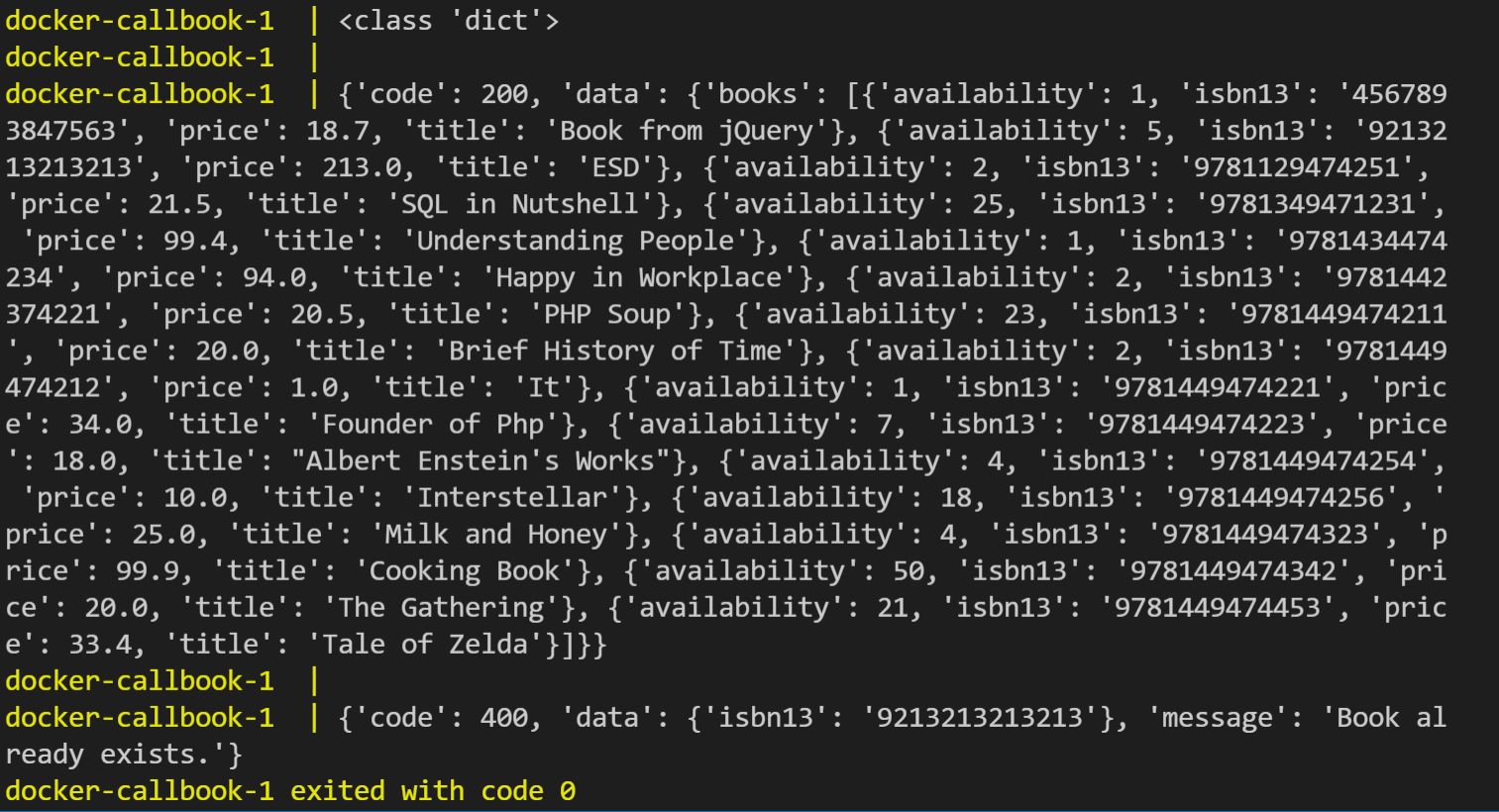
**Terminal → New Terminal**

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

Enter the following:

**docker-compose up**

****

****

*The following happens when you run docker-compose up:*

* *By default, docker-compose looks for the compose file named docker-compose.yml in the current folder. Use -f, --file FILE to specify an alternate compose file*
* *A default network named <projectname>\_default is created (docker is the name of the folder where the compose file lives and is used as the projectname and prefix to any names compose creates. Use -p, --project-name NAME to specify an alternate project name)*
* *A container named docker-book-1 is created. It joins the docker\_default network under the name book.*

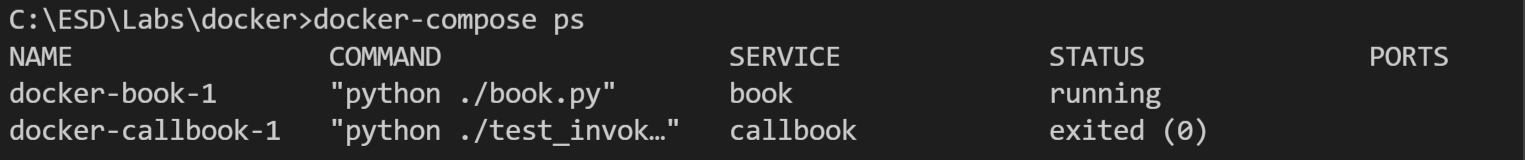
*The postfix 1 indicates the instance of the container.*

* *A container named docker-callbook-1 is created. It joins the docker\_default network under the name callbook.*
* *The output of the containers (prefixed with each container name) will also be printed*

1. Create a new terminal and enter the following:

(change directory to where you have your docker-compose.yml file)

**docker-compose ps**



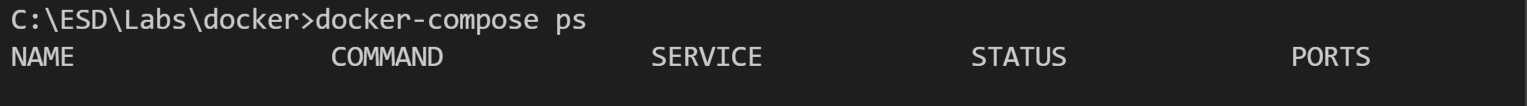
*You will see the state of the 2 containers. In this case, book is still running while callbook has stopped.*

1. Bring down everything by entering the following:

**docker-compose down**

****

*When you run docker-compose ps again, you will see that there are no containers (stopped or otherwise)*

**

*When you run docker ps, you will also see that there are no traces of these 2 containers (stopped or otherwise)*

*In future labs, you will apply what you have learned here to manage more microservices and containers we’ll create for our bookstore.*

# Learning Points

* *Run multi-container Docker applications with Compose*
* *Create compose file*
* *Work with Docker Compose commands and options (docker-compose up, ps, down)*