

# Docker Compose for Microfrontend

In this tutorial, we will cover how you can use Docker, specifically Docker Compose to more easily manage your microfrontends and run them seamlessly.

You can follow along using the code from [this](#) repository.

Docker is a platform that enables developers to package, distribute, and run applications in lightweight, portable containers. These containers encapsulate all the dependencies and settings required for an application, ensuring consistent and efficient deployment across different environments.

## Prerequisites

- [Docker Desktop](#)
- Nodejs
- VSCode

Before we begin, start by creating 3 microservices in one folder: a books app, a authors app and a main app which will serve as the gateway for the services.

### 1. Create Dockerfile for each of the microservices.

The first step when wanting to encapsulate a project with Docker is to create the Dockerfile. To do so, navigate to each project and create the file. Then populate it like so:

```
# Use an official Node.js runtime as a parent image
FROM node:14

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy package.json and package-lock.json to the working directory
COPY package*.json ./

# Install app dependencies
RUN npm install

# Copy the rest of the application code to the working directory
COPY . .

# Expose the port on which the application will run
```

```
EXPOSE 3003

# Define the command to run your application
CMD ["npm", "start"]
```

For this working example, you can use the code above. Just be sure to set a different port for each of the microservices in the EXPOSE line.

## 2. Create the Docker Compose file

This file serves to specify information about the entry point into your app and allows you to run all microservices with one command.

Create a file called docker-compose.yml in the root of the folder and populate it like so:

```
version: '3'

services:
  books-app:
    build: ./books-app
    ports:
      - "3003:3003"

  authors-app:
    build: ./authors-app
    ports:
      - "3002:3002"

  main-app:
    build: ./main-app
    ports:
      - "3000:3000"
    depends_on:
      - books-app
      - authors-app
```

As you can see, this code tells Docker which microservices to run and on which ports. Be sure that the ports correspond to the ones you set in each microservices' Dockerfile.

## 3. Build and Run Docker

Now that we've setup the files we need, it's time to build the Docker container. In order to do so, make sure you have the Docker app opened and in the root of your project, run:

```
docker-compose build
```

Then, once the command finishes successfully, run:

```
docker-compose up
```


Now, in your Docker app, you will be able to see your container. Moreover, your microservices are now running. All you need to do is navigate to localhost:3000 (or whatever port you set for the main app) and watch your app in action.


**Containers** [Give feedback](#)

Container CPU usage ⓘ  
*No containers are running.*

Container memory usage ⓘ  
*No containers are running.*

[Show charts](#)

 ☒ Only show running containers

<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	>  frontend		Running (3/3)	N/A		0 seconds ago	