

# **Docker Journal: Part 2**

Zainul Malik

## **Introduction**

This is a tutorial and outline of the steps I followed to complete this assignment.. It will take you through all the steps that you need to complete challenge 3 and challenge 4 outlined in the assignment document. Along with this, I will provide helpful references, screenshots, and general tips that you can use to learn how to deploy and scale applications using Docker, demonstrating both the configuration of multiple service components and the handling of increased load through service scaling.

## **Prerequisites**

Before you start, ensure you have the following installed on your computer:

- **Git:** For version control and cloning repositories.
- **Docker:** The main tool for creating, deploying, and running containers.
- **GitHub Account:** For hosting your project code and Dockerfiles.
- **Visual Studio Code:** A versatile code editor by Microsoft for Windows, macOS, and Linux.

## **Download Links**

These are the download links for the above programs:

- **Git:** <https://git-scm.com/downloads>
- **Docker:** <https://www.docker.com/products/docker-desktop/>
- **GitHub Account:** <https://github.com/>
- **Visual Studio Code:** <https://code.visualstudio.com/>

## **Helpful Links**

These are some helpful links I used to complete the challenges, and may be helpful for you as well.

- **Docker Docs:** <https://docs.docker.com/get-started/>
- **Creating a Dockerfile & Docker Image for HTML Application:** <https://youtu.be/UXAoZg1W3Q4?si=oriYwvfmHE6OVC5l>
- **NGINX + Docker:** <https://youtu.be/prn68HXjIwQ?si=ql9ceDd3x-Ca8CP3>
- **Understanding Docker Networking:** <https://www.docker.com/blog/understanding-docker-networking-drivers-use-cases/>
- **Docker Compose ENV File:** <https://www.youtube.com/watch?v=1je3VxDF67o>
- **Docker Docs ENV's:** <https://docs.docker.com/compose/environment-variables/>
- **Docker Compose Up:** <https://docs.docker.com/reference/cli/docker/compose/up/>
- **Scaling with Docker Compose:** <https://www.appsdeveloperblog.com/scaling-with-docker-compose-a-beginners-guide/>
- **Docker and MariaDB:** [https://hub.docker.com/\\_/mariadb](https://hub.docker.com/_/mariadb)

## **Getting Started**

1. **Fork the Project Repository:** Visit <https://github.com/eduluz1976/docker-challenge-template> and fork the repository to your GitHub account. Keep the forked repository private.
  - a. **NOTE:** In case you already submitted your first project (part 1), you can use your existing repository
2. **Clone the Repository:** Open your terminal and clone the forked repository to your local machine using:
  - a. `git clone <repository-url>`
  - b. Replace <repository-url> with the URL of your forked repository.

## **Folder Structure**

The repository folders should follow this structure:

- - README.md
- - student.cfg
- + challenge3
- + challenge4

The file student.cfg will contain the student information, and the README.md some information about your work. Use README.md to write your learnings during this project.

## **Goals**

### **Challenge 3: Full Stack Application Deployment**

- **Understand Inter-service Communication:** Learn how different services like web servers, applications, and databases interact within Docker.
- **Implement and Manage Configurations:** Use environment variables and docker-compose.yml to configure each service effectively.

### **Challenge 4: Scaling up an Application**

- **Demonstrate Scaling:** Scale the Node.js service from one to three instances to understand load distribution and the benefits of redundancy.
- **Evaluate Load Balancing:** Observe and document how Docker handles multiple service instances to ensure high availability and reliability.

These goals aim to enhance your practical knowledge of Docker in deploying and scaling complex applications.

# Challenge 3 – Full Stack Application

---

## Introduction

In this part of the journal, we will focus on building a full-stack application using Docker. This includes setting up a web server, a Node.js application, and a database. The goal is to demonstrate how these components interact within Docker and how to manage them using Docker Compose.

## Objectives

- Recognize and set up the ways in which various Docker services interact with one another.
- To safely configure the application components, use environment variables.
- In order to manage multi-container applications, create a docker-compose.yml file.

## Setup Steps

### 1. Prepare Your Environment:

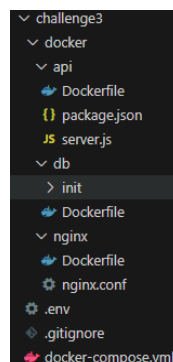
- a. Navigate to the “challenge3” folder
- b. If you have not already created it, you can do so by running: `mkdir challenge3`

```
C:\Users\zainu\Desktop\OS_BONUS_ASSIGNMENT\docker-challenge-template>mkdir challenge3
```

2. Once the challenge3 file has been created, navigate into it using: `cd challenge3`

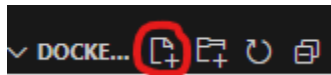
```
C:\Users\zainu\Desktop\OS_BONUS_ASSIGNMENT\docker-challenge-template>cd challenge3
```

3. Add the necessary files from the “challenge3.zip” provided on D2L.
  - a. Extract all files from the “challenge3.zip” into the “challenge3” directory. Make sure that you have the following structure within the “docker” folder:
    - i. `nginx` folder containing your Nginx Docker configuration and site configuration files.
    - ii. `node-service` folder containing your Node.js application files.
    - iii. `db` folder containing your database Dockerfile and any scripts.



#### 4. Create Environment Configuration:

- a. Create a `.env` file in the root of `challenge3` to store your environment variables:
  - i. You can use the new file button located within Visual Studio Code if you wish



- b. Add the following code to your `.env` file. You can choose the values that you want to set for your environmental variables. The following is just an example:

```
.env M X
challenge3 > .env
1  MYSQL_ROOT_PASSWORD=password123
2  MYSQL_DATABASE=booksDB
3  MYSQL_USER=dbUser
4  MYSQL_PASSWORD=password123
5
6
7  DB_HOST=db
8  DB_DATABASE=booksDB
9  DB_USERNAME=dbUser
10 DB_PASSWORD=password123
```

#### 5. Construct the Docker Compose File:

- a. Create a `docker-compose.yml` file in the root of the `challenge3` directory, using the same button for adding a file that was shown in step 4. It should define three services: `nginx`, `node-service`, and `db`. Here is an example of the code you can use:

```
docker-compose.yml X
challenge3 > docker-compose.yml
1  version: '3.8'
2  services:
3    db:
4      build:
5        context: ../docker/db
6      environment:
7        MYSQL_ROOT_PASSWORD: "password123"
8        MYSQL_DATABASE: "booksDB"
9        MYSQL_USER: "dbUser"
10       MYSQL_PASSWORD: "password123"
11     ports:
12       - "3306:3306"
13     networks:
14       - app-network
15
16   node-service:
17     build:
18       context: ../docker/api
19     ports:
20       - "3000:3000"
21     depends_on:
22       - db
23     environment:
24       DB_HOST: db
25       DB_DATABASE: "booksDB"
26       DB_USERNAME: "dbUser"
27       DB_PASSWORD: "password123"
28     networks:
29       - app-network
30
31   nginx:
32     build:
33       context: ../docker/nginx
34     ports:
35       - "8080:80"
36     depends_on:
37       - node-service
38     networks:
39       - app-network
40
41   networks:
42     app-network:
43       driver: bridge
```

## 6. Launch Your Application:

- a. From the `challenge3` directory, run: `docker-compose up --build`. This command builds the images for your services and starts the containers.

```
C:\Users\zainu\Desktop\OS_BONUS_ASSIGNMENT\docker-challenge-template\challenge3>docker-compose up --build
```

- b. After running this command, you should see a similar output:

[illegible]

## 7. Verify the Application:

- a. Open a browser and visit <http://localhost:8888/api/books> to see if you get a JSON list of books. The result should look something like this:

localhost:3000/api/books

JSON Raw Data Headers

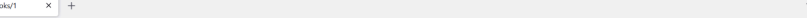
Save Copy Collapse All Expand All Filter JSON

```

{
  "id": 1,
  "title": "To Kill a Mockingbird",
  "author": "Harper Lee"
},
{
  "id": 2,
  "title": "1984",
  "author": "George Orwell"
},
{
  "id": 3,
  "title": "Pride and Prejudice",
  "author": "Jane Austen"
},
{
  "id": 4,
  "title": "The Great Gatsby",
  "author": "F. Scott Fitzgerald"
}

```

- b. Check <http://localhost:8080/api/books/1> to get details on a specific book. The result should look something like this:



The screenshot shows a web browser window with the address bar displaying `localhost:8080/api/books/1`. The page content shows a JSON response with the following fields:

```

{
  "id": 1,
  "title": "To Kill a Mockingbird",
  "author": "Harper Lee"
}

```

- c. If there are issues, check your Docker logs and configurations, then adjust accordingly.

## 8. Document Your Process and Commit Your Work:

- a. Commit all files and push them to the remote repository using the following commands in this order:
  - i. `git add .`
  - ii. `git commit -m 'Your commit message'`
  - iii. `git push origin master`
- b. Add screenshots of your terminal, browser, and command prompt showing the commands you used, and the results.
- c. Capture the output of `docker-compose ps` showing all services running correctly similar to this:

```
C:\Users\zainu\Desktop\OS_BONUS_ASSIGNMENT\docker-challenge-template\challenge3>docker-compose ps
NAME                IMAGE                COMMAND                SERVICE    CREATED        STATUS        PORTS
challenge3-db-1      challenge3-db        "docker-entrypoint.s..." db          6 seconds ago  Up 5 seconds  0.0.0.0:3306->3306/tcp
challenge3-nginx-1   challenge3-nginx     "/docker-entrypoint..." nginx       6 seconds ago  Up 5 seconds  0.0.0.0:8080->80/tcp
challenge3-node-service-1 challenge3-node-service "docker-entrypoint.s..." node-service 6 seconds ago  Up 5 seconds  0.0.0.0:3000->3000/tcp
```

- d. Note down all commands used and describe your troubleshooting steps, if any.

## Conclusion

This tutorial walked you through setting up a Docker environment for a full-stack application. You learned how to manage multiple services, utilize environment variables for configuration, and deploy a full application using Docker Compose.

# Challenge 4 - Scaling Up An Application

## Introduction

In this challenge, you will learn how to scale a service within a Docker Compose environment. The node service from Challenge 3 will be used as the basis for this scaling exercise. Our goal is to increase the number of instances of the node service from 1 to 3, and then observe how this affects the distribution of requests among these instances.

## Goals

- Increase the number of node instances from one to three.
- Recognize the advantages and the results of expanding a service inside a Dockerized application.

## Prerequisites

- Complete Challenge 3 to ensure that your Docker Compose setup is working correctly with one instance of each service.

## Steps to Scale Up the Service

### 1. Initial Setup and Testing:

- Start with the application setup from Challenge 3.
- Make multiple GET requests to <http://localhost:8080/api/stats> and record the **hostname** to confirm that it remains consistent across requests. This is to confirm that only one instance is handling all the requests. An example result can be seen below:



### 2. Scaling the Node Service:

- To scale up the node service to three instances, use the following Docker Compose command: `docker-compose up --scale node-service=3 -d`. You can search up how to scale up a node service using the link related to scaling in the “Helpful Links” section.
- This command tells Docker Compose to scale the **node-service** to 3 instances and run it in detached mode.

```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\zainu\Desktop\OS_BONUS_ASSIGNMENT\docker-challenge-template\challenge3>docker-compose up --scale node-service=3 -d
[+] Running 5/5
✓Container challenge3-db-1           Running      0.0s
✓Container challenge3-node-service-3 Started      11.5s
✓Container challenge3-node-service-2 Started      11.8s
✓Container challenge3-node-service-1 Started      12.1s
✓Container challenge3-nginx-1        Started      1.9s
```



### 3. Testing Post-Scaling:

- a. After scaling, repeat the GET requests to <http://localhost:8080/api/stats>. Now, you should see different **hostname** values for the responses. This indicates that different instances are handling the requests. This change demonstrates the load balancing (typically Round-Robin) between the instances. Since we have 3 instances running, you should see 3 unique hostnames that you **must note down**. Examples are shown below:



#### 4. Document the Evidence and Commit Your Work:

- Commit all files and push them to the remote repository using the following commands in this order:
  - `git add .`
  - `git commit -m 'Your commit message'`
  - `git push origin master`
- Add screenshots of your terminal, browser, and command prompt showing the commands you used, and the results.
- Capture the output of `docker-compose ps` showing all services running correctly similar to this:

```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\zainu\Desktop\OS_BONUS_ASSIGNMENT\docker-challenge-template\challenge3>docker-compose up --scale node-service=3 -d
[+] Running 5/5
 ✓ Container challenge3-db-1      Running      0.0s
 ✓ Container challenge3-node-service-3 Started      11.5s
 ✓ Container challenge3-node-service-2 Started      11.8s
 ✓ Container challenge3-node-service-1 Started      12.1s
 ✓ Container challenge3-nginx-1   Started      1.9s

C:\Users\zainu\Desktop\OS_BONUS_ASSIGNMENT\docker-challenge-template\challenge3>docker-compose ps
NAME                IMAGE                COMMAND                  SERVICE    CREATED   STATUS    PORTS
challenge3-db-1     challenge3-db        "docker-entrypoint.s..." db         21 hours ago Up 21 hours 0.0.0.0:3306->3306/tcp
challenge3-nginx-1 challenge3-nginx      "/docker-entrypoint..." nginx      8 minutes ago Up 8 minutes 0.0.0.0:8080->80/tcp
challenge3-node-service-1 challenge3-node-service "docker-entrypoint.s..." node-service 8 minutes ago Up 8 minutes 0.0.0.0:49683->3000/tcp
challenge3-node-service-2 challenge3-node-service "docker-entrypoint.s..." node-service 8 minutes ago Up 8 minutes 0.0.0.0:49681->3000/tcp
challenge3-node-service-3 challenge3-node-service "docker-entrypoint.s..." node-service 8 minutes ago Up 8 minutes 0.0.0.0:49680->3000/tcp
```

- Note down all commands used and describe your troubleshooting steps, if any.

### Expected Outcomes

- Before Scaling:** A **single hostname** response from multiple requests, showing no change across requests.
- After Scaling:** **Multiple hostname** responses from similar requests, showing that the load is being distributed among three instances.

### Conclusion

Scaling the node service demonstrates a core capability of Docker which is supporting high availability and load balancing within microservices architecture. This simple exercise will help you understand the practical effects of scaling and its benefits, such as improved fault tolerance and response times.

# Glossary

---

## 1. Docker

An open-source platform that uses containerization technology to enable developers to package applications with their dependencies and deploy them as one package.

- a. **Reference:** "Docker Overview," *Docker Docs*. Available: <https://docs.docker.com/get-started/overview/>.

## 2. Git

A version control system for tracking changes in computer files and coordinating work on those files among multiple people.

- a. **Reference:** "About Git," *Git*. Available: <https://git-scm.com/about>.

## 3. GitHub

A cloud-based hosting service that lets you manage Git repositories. It's used for version control and collaborative development.

- a. **Reference:** "What is GitHub?," *GitHub Docs*. Available: <https://docs.github.com/en/github/getting-started-with-github/github-glossary#github>.

## 4. Visual Studio Code (VS Code)

A lightweight but powerful source code editor from Microsoft, available for Windows, macOS, and Linux. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other programming languages.

- a. **Reference:** "Visual Studio Code Docs," *Visual Studio Code*. Available: <https://code.visualstudio.com/docs>.

## 5. Docker Compose

A tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services, networks, and volumes, and then create and start all the services from your configuration with a single command.

- a. **Reference:** "Overview of Docker Compose," *Docker Docs*. Available: <https://docs.docker.com/compose/>.

## 6. Container

A lightweight, stand-alone, executable package of software that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and configuration files.

- a. **Reference:** "Docker Glossary - Container," *Docker Docs*. Available: <https://docs.docker.com/glossary/?term=container>.

## 7. **Image**

In Docker, an image is a read-only template used to create containers. Containers are instances of Docker images that can be run using the Docker run command.

- a. **Reference:** "Docker Glossary - Image," *Docker Docs*. Available: <https://docs.docker.com/glossary/?term=image>.

## 8. **Dockerfile**

A text document that contains all the commands a user could call on the command line to assemble an image. Using docker build, users can create an automated build that executes several command-line instructions in succession.

- a. **Reference:** "Dockerfile reference," *Docker Docs*. Available: <https://docs.docker.com/engine/reference/builder/>.

## 9. **Environment Variables**

Key-value pairs that can be set for a container or service in a shell environment and are used to affect the way running processes will behave on a computer.

- a. **Reference:** "Environment variables in Compose," *Docker Docs*. Available: <https://docs.docker.com/compose/environment-variables/>.

## 10. **Load Balancing**

The process of distributing network or application traffic across multiple servers. In Docker, this is typically used to distribute the load of your application across multiple instances to increase redundancy and reliability.

- a. **Reference:** "Docker Swarm Load Balancing," *Docker Docs*. Available: <https://docs.docker.com/engine/swarm/ingress/>.

## 11. **Microservices**

A style of software architecture where complex applications are composed of small, independent processes communicating with each other using language-agnostic APIs.

- a. **Reference:** "Microservices," *Microservices.io*. Available: <https://microservices.io/>.

## 12. **YAML**

A human-readable data serialization standard that is commonly used for configuration files and in applications where data is being stored or transmitted.

- a. **Reference:** "YAML," *YAML.org*. Available: <https://yaml.org/spec/>.

## References

---

- [1] Docker Docs, "Get Started with Docker," Docker, 2024. [Accessed: 22-Apr-2024]. [Online]. Available: <https://docs.docker.com/get-started/>
- [2] "Creating a Dockerfile & Docker Image for HTML Application," YouTube, 2024. [Accessed: 22-Apr-2024]. [Online]. Available: <https://youtu.be/UXAoZg1W3Q4?si=oriYwvfmHE6OVC5l>
- [3] "NGINX + Docker," YouTube, 2024. [Accessed: 22-Apr-2024]. [Online]. Available: <https://youtu.be/prn68HXjIwQ?si=ql9ceDd3x-Ca8CP3>
- [4] Docker, "Understanding Docker Networking," Docker Blog, 2024. [Accessed: 22-Apr-2024]. [Online]. Available: <https://www.docker.com/blog/understanding-docker-networking-drivers-use-cases/>
- [5] "Docker Compose ENV File," YouTube, 2024. [Accessed: 22-Apr-2024]. [Online]. Available: <https://www.youtube.com/watch?v=lje3VxDF67o>
- [6] Docker Docs, "Environment Variables in Compose," Docker, 2024. [Accessed: 22-Apr-2024]. [Online]. Available: <https://docs.docker.com/compose/environment-variables/>
- [7] Docker Docs, "Compose Up Command," Docker, 2024. [Accessed: 22-Apr-2024]. [Online]. Available: <https://docs.docker.com/reference/cli/docker/compose/up/>
- [8] Apps Developer Blog, "Scaling with Docker Compose: A Beginner's Guide," Apps Developer Blog, 2024. [Accessed: 22-Apr-2024]. [Online]. Available: <https://www.appsdeveloperblog.com/scaling-with-docker-compose-a-beginners-guide/>
- [9] Docker Hub, "MariaDB on Docker," Docker Hub, 2024. [Accessed: 22-Apr-2024]. [Online]. Available: [https://hub.docker.com/\\_/mariadb](https://hub.docker.com/_/mariadb)