Challenge 3&4 Journal

Setup and Configuration

Docker Environment Setup

Docker and Docker Compose are utilized to containerize and manage our multi-tier application. Docker Compose handles multiple containers to work together as a single service.

Repository Structure

Fork the Docker Challenge repository to begin. Use the challenge3.zip provided to copy all folders to the root of challenge3 folder. It should look like this afterwards

```
docker-challenge-template
-- challenge1/
-- challenge2/
-- challenge3/
        -- api/
               Dockerfile
               package.json
               server.js
        -- db/
               init/
                       init.sql
               Dockerfile
        -- nginx
               Dockerfile
               nginx.conf
-- challenge4/
```

Creating Configuration Files

To configure the multi-container setup functions correctly, we need to create two essential configuration files: '.env' and 'docker-compose.yml'. These files will store environment settings and define how the Docker containers interact.

Creating the '.env' File

The .env file contains environment variables that are crucial for the configuration of our services, particularly the database credentials and connection settings. Here's how to create and populate this file:

1. Navigate to the 'challenge3' directory. `cd challenge3` if inside the docker-template folder
2. Create a new file named '.env'
3. Edit the '.env' file to add the following content.

```
# Database Configuration for MariaDB
MYSQL_ROOT_PASSWORD=choose_a_secure_password
MYSQL_DATABASE=books_db
MYSQL_USER=books_user
MYSQL_PASSWORD=another_secure_password
MYSQL_HOST=db

# Node App Configuration
DB_HOST=${MYSQL_HOST}
DB_USERNAME=${MYSQL_USER}
DB_PASSWORD=${MYSQL_PASSWORD}
DB_DATABASE=${MYSQL_DATABASE}
```

Replace the passwords with ones of your choosing.

Creating the Docker Compose File

docker-compose.yml orchestrates our Docker containers, defining how they are built and how they interact with each other:

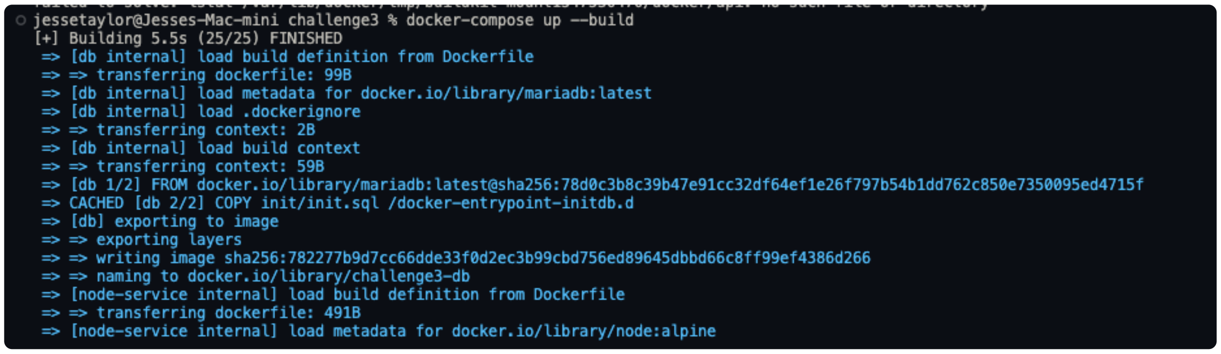
1. Navigate to the 'challenge3' directory if you are not still there.
2. Create a new file named 'docker-compose.yml'.
3. Edit the 'docker-compose.yml' file to include these configurations. Refer to this page if you're facing difficulties.

```
version: '3.8'
services:
  db:
    build: ./db
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    volumes:
      - db_data:/var/lib/mysql

  node-service:
    build: ./api
    depends_on:
      - db
    environment:
      DB_HOST: ${DB_HOST}
      DB_USERNAME: ${DB_USERNAME}
      DB_PASSWORD: ${DB_PASSWORD}
      DB_DATABASE: ${DB_DATABASE}
    ports:
      - "3000:3000"

  nginx:
    build: ./nginx
    depends_on:
      - node-service
    ports:
      - "8080:80"

volumes:
  db_data:
```

Running our Application

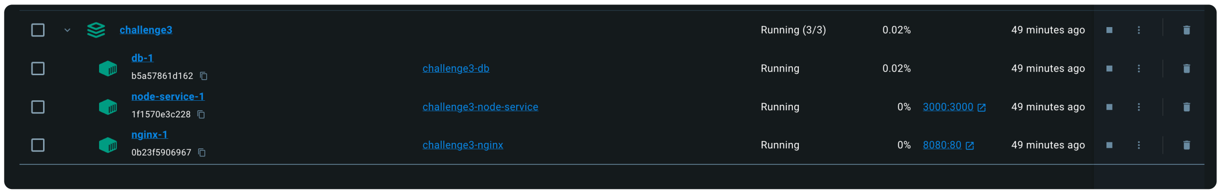
Once these two files have been created and properly connect our multi-container application together. We can run this application by doing the following:

1. Once again, navigate to the challenge3 directory. `cd challenge3`
2. Execute the following command: `docker-compose up --build`
    1. Command in terminal:



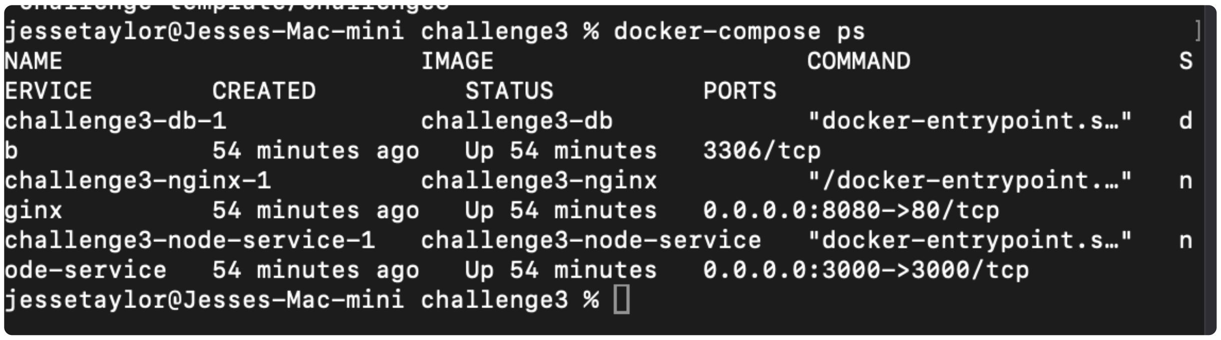
```
○ jessetaylor@Jesses-Mac-mini challenge3 % docker-compose up --build
  [+] Building 5.5s (25/25) FINISHED
   => [db internal] load build definition from Dockerfile
   => => transferring dockerfile: 99B
   => [db internal] load metadata for docker.io/library/mariadb:latest
   => [db internal] load .dockerignore
   => => transferring context: 2B
   => [db internal] load build context
   => => transferring context: 59B
   => [db 1/2] FROM docker.io/library/mariadb:latest@sha256:78d0c3b8c39b47e91cc32df64ef1e26f797b54b1dd762c850e7350095ed4715f
   => CACHED [db 2/2] COPY init/init.sql /docker-entrypoint-initdb.d
   => [db] exporting to image
   => => exporting layers
   => => writing image sha256:782277b9d7cc66dde33f0d2ec3b99cbd756ed89645dbbd66c8ff99ef4386d266
   => => naming to docker.io/library/challenge3-db
   => [node-service internal] load build definition from Dockerfile
   => => transferring dockerfile: 491B
   => [node-service internal] load metadata for docker.io/library/node:alpine
```

2. Application running in Docker:



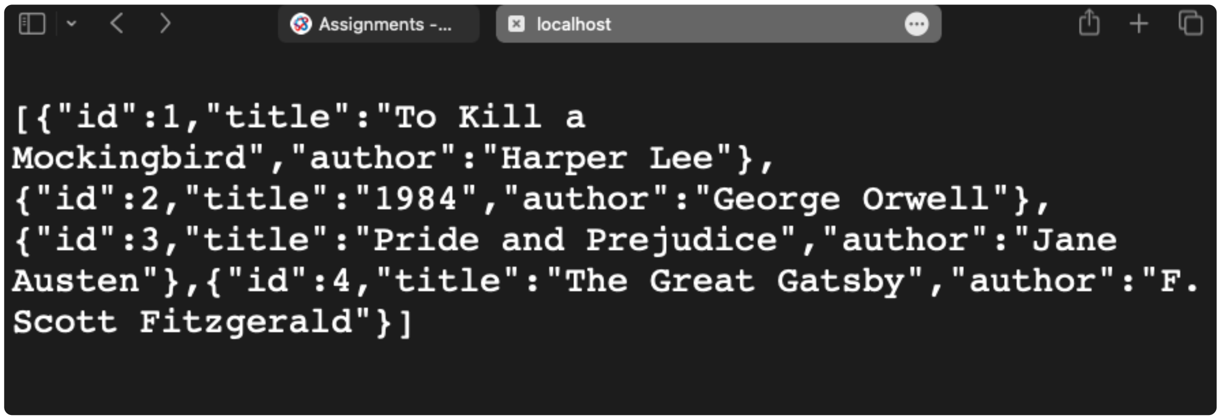
challenge3		Running (3/3)	0.02%		49 minutes ago
db-1 b5a57861d162	challenge3-db	Running	0.02%		49 minutes ago
node-service-1 1f1570e3c228	challenge3-node-service	Running	0%	3000:3000	49 minutes ago
nginx-1 0b23f5906967	challenge3-nginx	Running	0%	8080:80	49 minutes ago

3. Verify Docker is running correctly
   1. Command in terminal: `docker-compose ps`



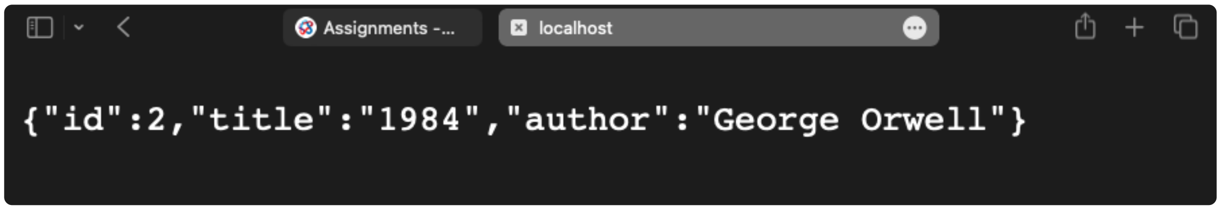
```
jessetaylor@Jesses-Mac-mini challenge3 % docker-compose ps
NAME                        IMAGE                     COMMAND                    S
ERVICE         CREATED          STATUS           PORTS
challenge3-db-1             challenge3-db             "docker-entrypoint.s…"     d
b              54 minutes ago   Up 54 minutes    3306/tcp
challenge3-nginx-1          challenge3-nginx          "/docker-entrypoint.…"     n
ginx           54 minutes ago   Up 54 minutes    0.0.0.0:8080->80/tcp
challenge3-node-service-1   challenge3-node-service   "docker-entrypoint.s…"     n
ode-service    54 minutes ago   Up 54 minutes    0.0.0.0:3000->3000/tcp
jessetaylor@Jesses-Mac-mini challenge3 % 
```

   2. Verify API Books, access the following URL in a web browser. `http://localhost:8080/api/books` and you should see the following:



```
[{"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"}]
```

   3. Verify fetching a specific book. `http://localhost:8080/api/books/2`:



```
{"id":2,"title":"1984","author":"George Orwell"}
```

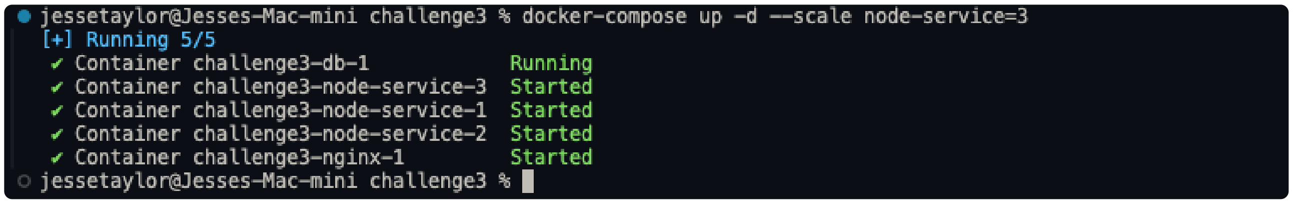
We have now verified our docker application is working in all aspects.

Challenge 4 Scaling up

Scaling Procedure

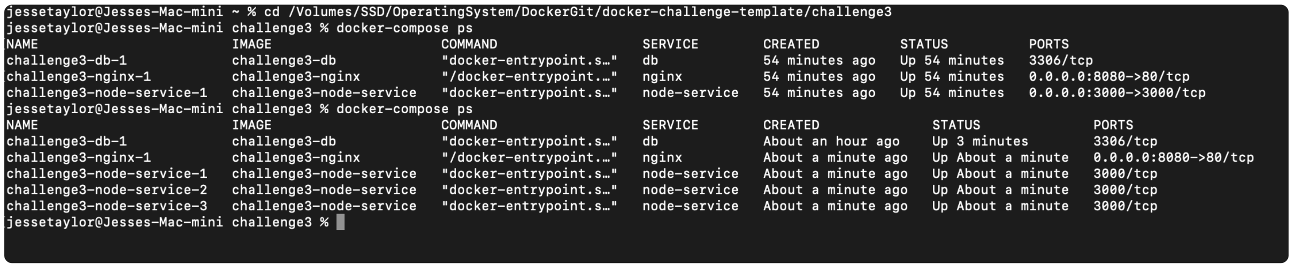
Since our .yml is already scalable, we can do the following:

1. Scale the 'node-service' to 3 instances using: `docker-compose up -d --scale node-service=3`



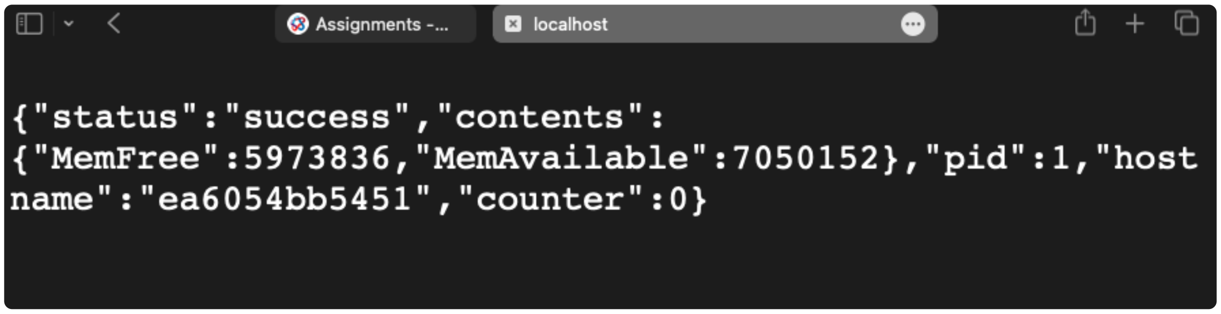
```
jessetaylor@Jesses-Mac-mini challenge3 % docker-compose up -d --scale node-service=3
 [+] Running 5/5
  ✔ Container challenge3-db-1              Running
  ✔ Container challenge3-node-service-3    Started
  ✔ Container challenge3-node-service-1    Started
  ✔ Container challenge3-node-service-2    Started
  ✔ Container challenge3-nginx-1           Started
jessetaylor@Jesses-Mac-mini challenge3 %
```

2. Confirm the scaling with `docker-compose ps`

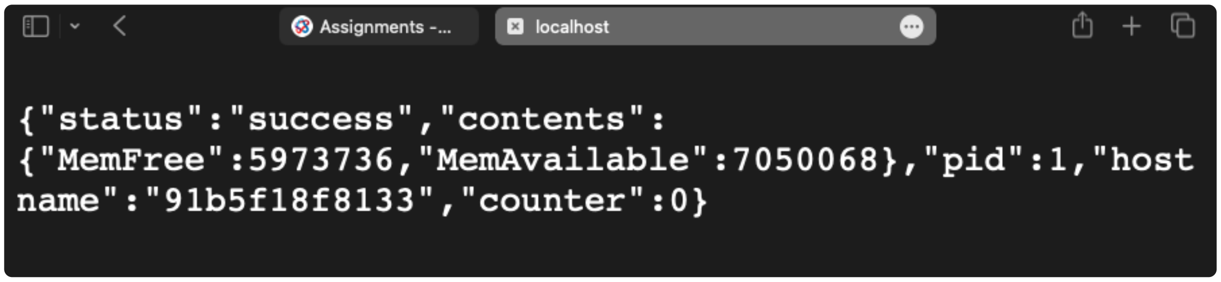


```
jessetaylor@Jesses-Mac-mini ~ % cd /Volumes/SSD/OperatingSystem/DockerGit/docker-challenge-template/challenge3
jessetaylor@Jesses-Mac-mini challenge3 % docker-compose ps
NAME                          IMAGE                       COMMAND                    SERVICE          CREATED          STATUS          PORTS
challenge3-db-1               challenge3-db               "docker-entrypoint.s…"     db               54 minutes ago   Up 54 minutes   3306/tcp
challenge3-nginx-1            challenge3-nginx            "/docker-entrypoint.…"     nginx            54 minutes ago   Up 54 minutes   0.0.0.0:8080->80/tcp
challenge3-node-service-1     challenge3-node-service     "docker-entrypoint.s…"     node-service     54 minutes ago   Up 54 minutes   0.0.0.0:3000->3000/tcp
jessetaylor@Jesses-Mac-mini challenge3 % docker-compose ps
NAME                          IMAGE                       COMMAND                    SERVICE          CREATED              STATUS              PORTS
challenge3-db-1               challenge3-db               "docker-entrypoint.s…"     db               About an hour ago    Up 3 minutes        3306/tcp
challenge3-nginx-1            challenge3-nginx            "/docker-entrypoint.…"     nginx            About a minute ago   Up About a minute   0.0.0.0:8080->80/tcp
challenge3-node-service-1     challenge3-node-service     "docker-entrypoint.s…"     node-service     About a minute ago   Up About a minute   3000/tcp
challenge3-node-service-2     challenge3-node-service     "docker-entrypoint.s…"     node-service     About a minute ago   Up About a minute   3000/tcp
challenge3-node-service-3     challenge3-node-service     "docker-entrypoint.s…"     node-service     About a minute ago   Up About a minute   3000/tcp
jessetaylor@Jesses-Mac-mini challenge3 %
```

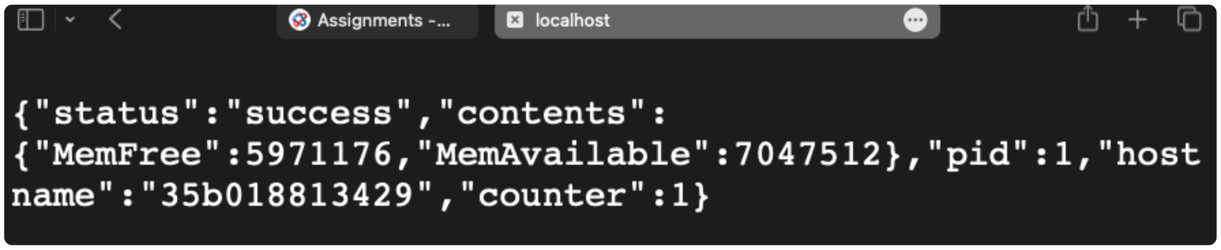
Verifying Load Balancing

1. Make multiple GET requests to `http://localhost:8080/api/stats` and record the different hostnames to demonstrate round-robin load balancing.
   1. 

```
{"status":"success","contents":{"MemFree":5973836,"MemAvailable":7050152},"pid":1,"hostname":"ea6054bb5451","counter":0}
```

   2. 

```
{"status":"success","contents":{"MemFree":5973736,"MemAvailable":7050068},"pid":1,"hostname":"91b5f18f8133","counter":0}
```

3. 

```
{"status":"success","contents":
{"MemFree":5971176,"MemAvailable":7047512},"pid":1,"host
name":"35b018813429","counter":1}
```