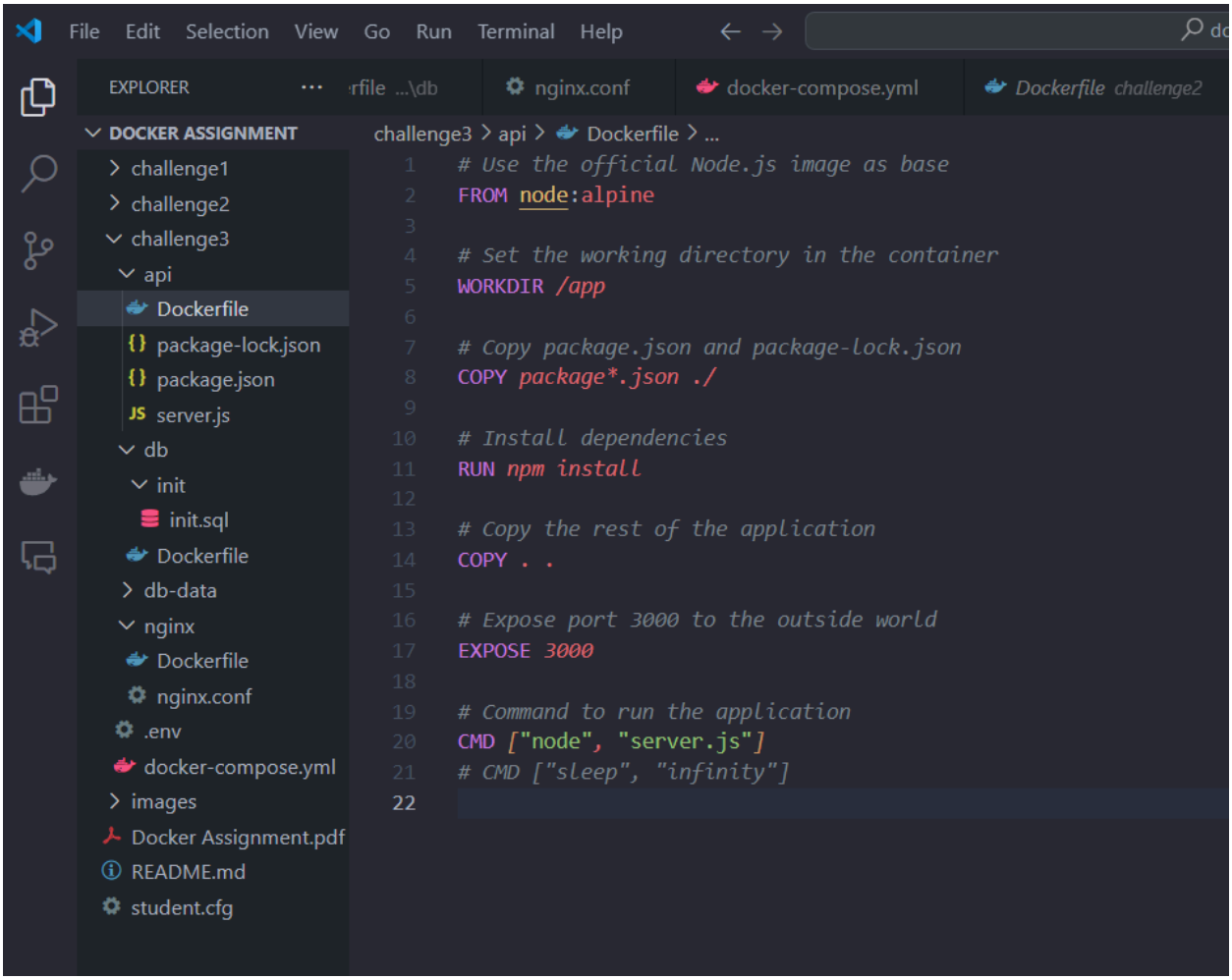
DOCKER FINAL

Jason Alfonza Kencana
000897961

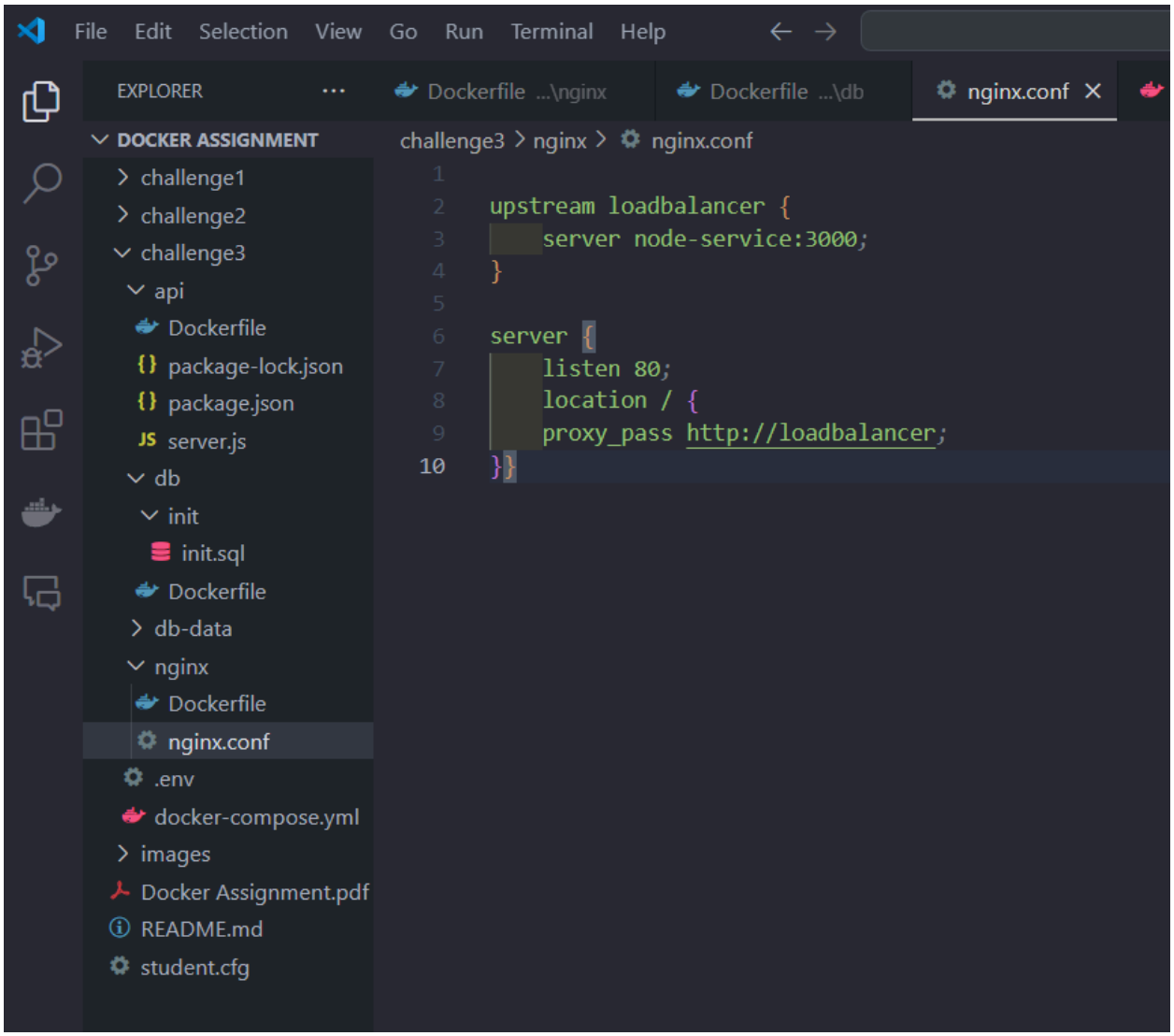
Dockerfile:



- FROM node:alpine
  - This line specifies the base image to use for building the Docker container, which is the Alpine version of the official Node.js image.
- WORKDIR /app
  - Sets the working directory inside the container where subsequent commands will be run
- COPY package*.json ./
  - Copies the package.json and package-lock.json files from the host machine to the working directory in the container.
- RUN npm install
  - Installs the dependencies listed in the package.json file using npm.
- COPY . .
  - Copies the rest of the application files from the host machine to the working directory in the container.

- EXPOSE 3000
    - Informs Docker that the container will listen on port 3000 at runtime.
- CMD ["node", "server.js"]
    - Specifies the command to run when the container starts. In this case, it runs the server.js file using Node.js. The commented out CMD command is an alternative approach to keep the container running indefinitely, useful for debugging purposes.

Nginx.conf:



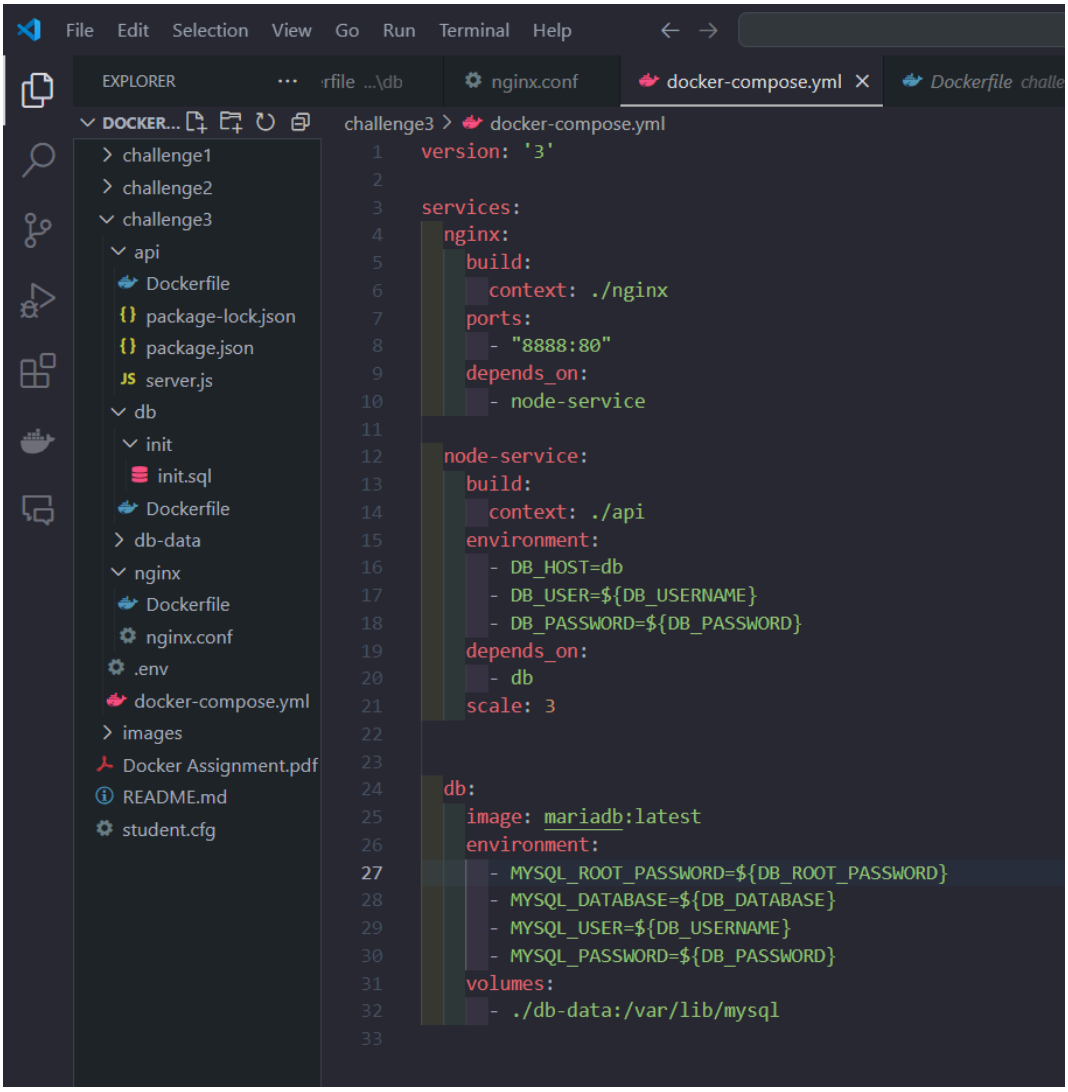
- upstream loadbalancer {
  server node-service:3000;
  }
    - Defines an upstream block named 'loadbalancer' with the address of the node-service container. This will be used by nginx to load balance requests.

- server {
    listen 80;
    location / {
        proxy_pass http://loadbalancer;
    }
}
    - Configures nginx to listen on port 80 and proxy incoming requests to the 'loadbalancer' upstream block, effectively load balancing the requests to the node-service containers.

.env:

- Contains environment variables used by the Docker Compose file, such as database credentials.

Docker-compose.yml:



- version: '3'

  services:
    nginx:
      build:
        context: ./nginx
      ports:
        - "8888:80"
      depends_on:
        - node-service

    node-service:
      build:
        context: ./api
      environment:
        - DB_HOST=db
        - DB_USERNAME=${DB_USERNAME}
        - DB_PASSWORD=${DB_PASSWORD}
        - DB_DATABASE=${DB_DATABASE}
      depends_on:
        - db
      scale: 3

    db:
      image: mariadb:latest
      environment:
        - MYSQL_ROOT_PASSWORD=${DB_ROOT_PASSWORD}
        - MYSQL_DATABASE=${DB_DATABASE}
        - MYSQL_USER=${DB_USERNAME}
        - MYSQL_PASSWORD=${DB_PASSWORD}
      volumes:
        - ./db-data:/var/lib/mysql

  - This Docker Compose file defines three services: nginx, node-service, and db.
    - The nginx service uses the Dockerfile located in the ./nginx directory to build the nginx container image. It exposes port 8888 on the host machine and depends on the node-service.
    - The node-service service uses the Dockerfile located in the ./api directory to build the node-service container image. It sets environment variables for database connection and scales the service to run 3 instances.

- The db service uses the mariadb:latest Docker image and sets environment variables for database configuration. It also mounts a volume for persistent data storage.

How to run the code (Challenge 3) :

1. Open a terminal
2. Navigate to the directory containing the Dockerfile
    a. In the assignment I provided these are the complete directory :
3. cd C:\docker\docker-challenge-template\challengefinal
4. Run “docker-compose up” on the terminal as it will build the docker images and start the containers
5. Now we can test it out by opening a web browser and navigating to http://localhost:8888/api/books, we can see the id, title, and author for all the books.
6. If we navigate to http://localhost:8888/api/books/1 or http://localhost:8888/api/books/2, http://localhost:8888/api/books/3 we can see each id, title, and the author for the specific book.

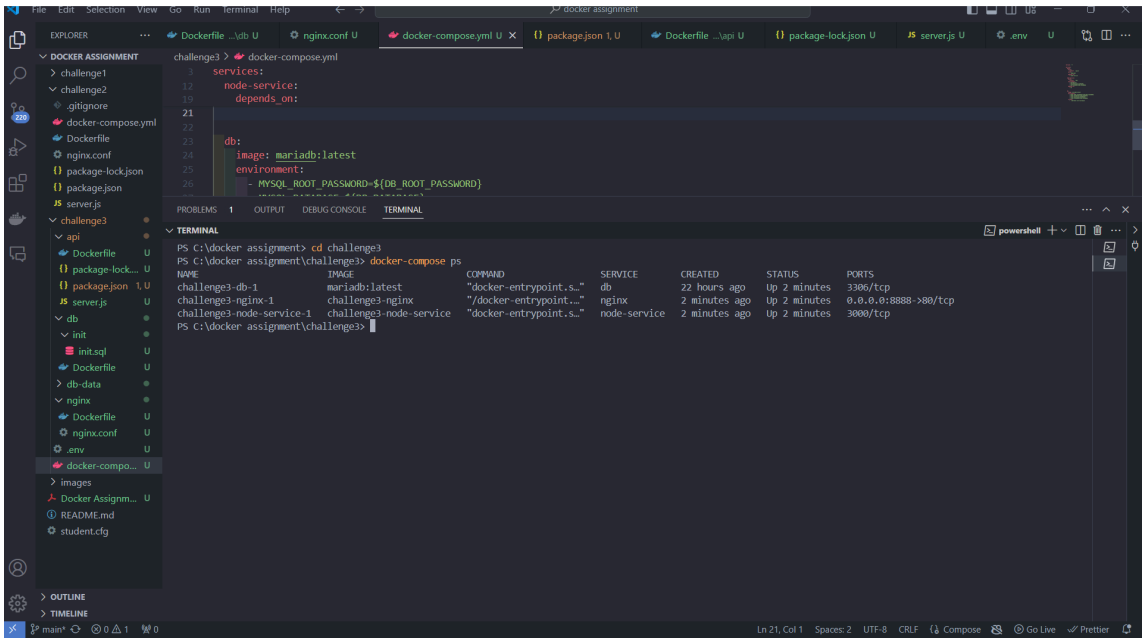
Challenge 4

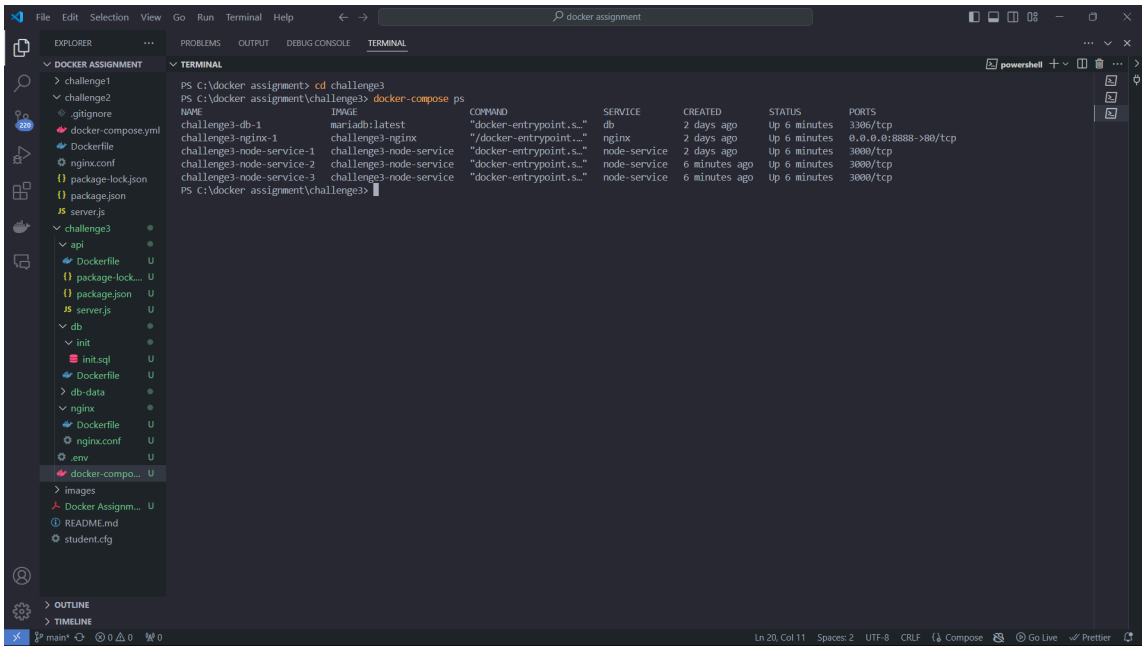
- Setup from Challenge 3:
    - The project began with using the setup from Challenge 3, which included a Docker Compose file defining three services: nginx, node-service, and db. This setup already had a working full-stack application with a web server, Node.js application, and database.
- Initial Testing:
    - Before scaling up the node service, initial testing was conducted by making GET requests to the application endpoint (http://localhost:8080/api/stats). The purpose was to record the hostname returned in the response and observe its consistency.
- Scaling Up:
    - The node service was scaled up from one instance to three instances by adding the scale: 3 configuration to the node-service in the Docker Compose file. This instructed Docker Compose to create and start three instances of the node-service container.

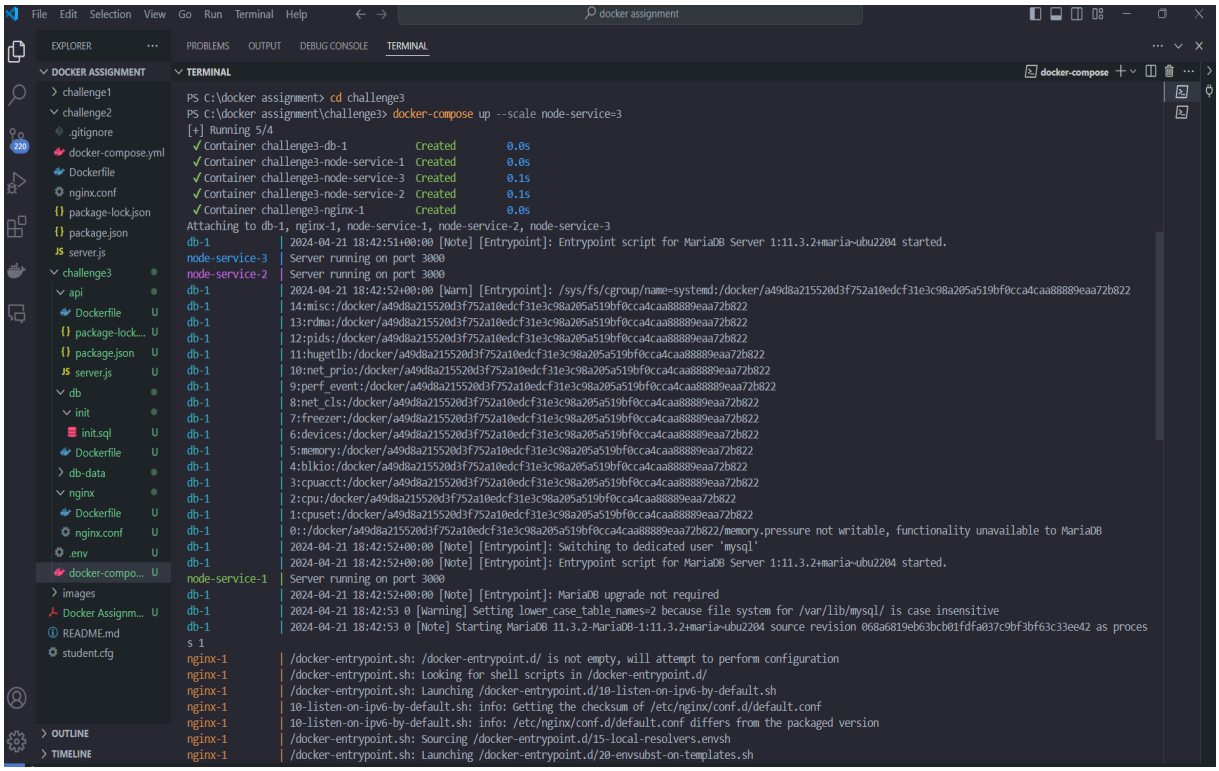
How to run the code (Challenge 4) :

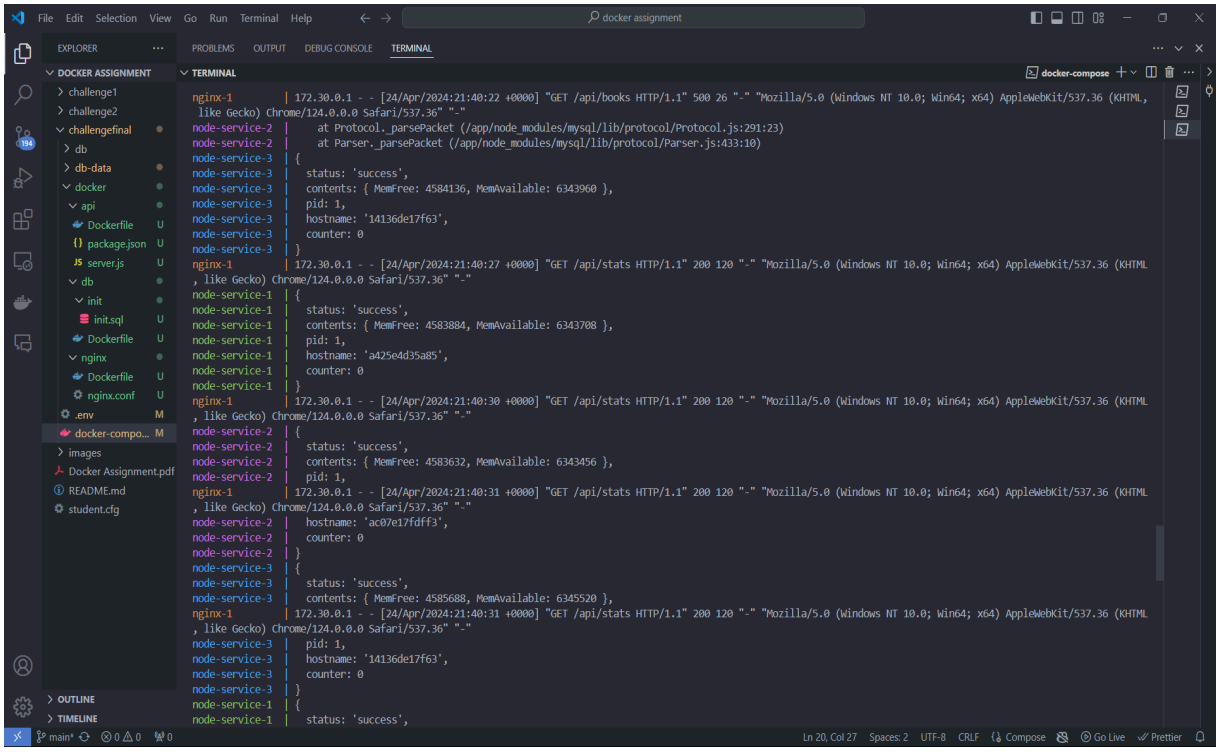
1. Open a terminal
2. Navigate to the directory containing the Dockerfile
    a. In the assignment I provided these are the complete directory :
3. cd C:\docker\docker-challenge-template\challengefinal

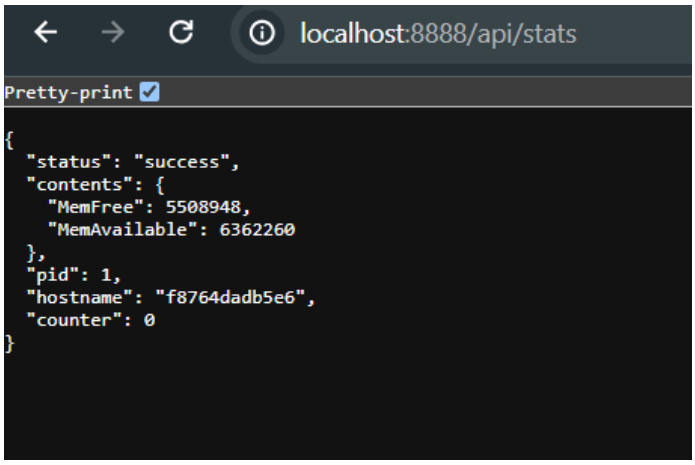
4. Run “docker-compose up --scale node-service=3” on the terminal as it will build the docker images and containers
5. After scaling up the node service, testing was repeated by making GET requests to the application endpoint multiple times. The objective was to record the hostname returned in each response and observe any differences compared to the initial testing.
6. We can also run “docker-compose ps” to show the status of all services defined in the docker-compose.yml file, including the number of running instances for each service.

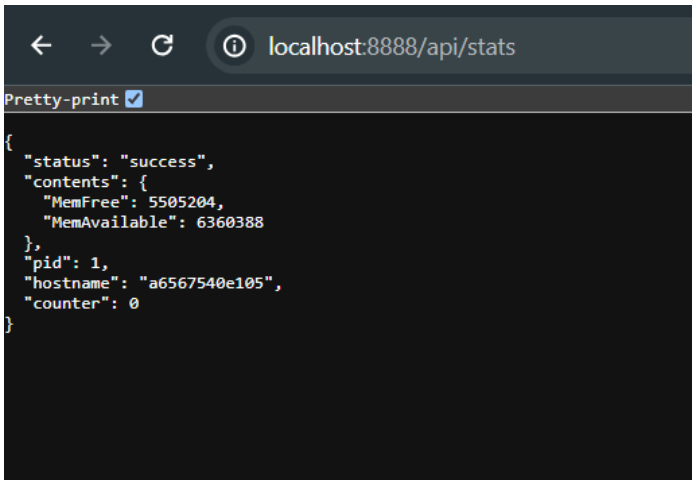


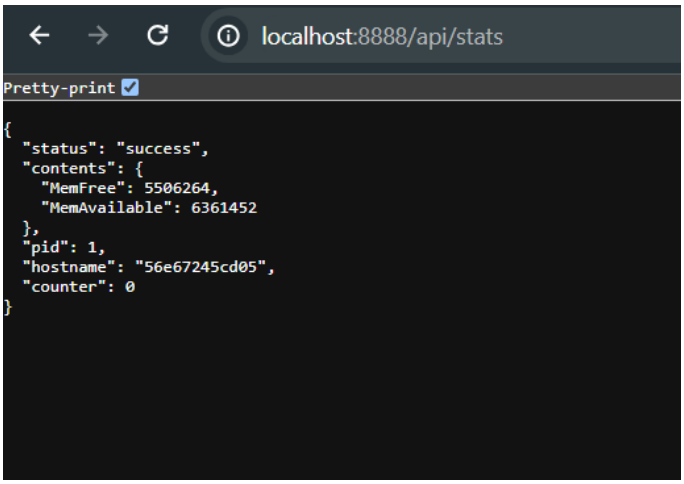



File Edit Selection View Go Run Terminal Help
docker assignment
EXPLORER
DOCKER ASSIGNMENT
challenge1
challenge2
.gitignore
docker-compose.yml
Dockerfile
nginx.conf
package-lock.json
package.json
server.js
challenge3
api
Dockerfile
package-lock....
package.json
server.js
db
init
init.sql
Dockerfile
db-data
nginx
Dockerfile
nginx.conf
.env
docker-compo...
images
Docker Assignm...
README.md
student.cfg
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
TERMINAL
docker-compose
PS C:\docker assignment> cd challenge3
PS C:\docker assignment\challenge3> docker-compose up --scale node-service=3
[+] Running 5/4
✓Container challenge3-db-1 Created 0.0s
✓Container challenge3-node-service-1 Created 0.0s
✓Container challenge3-node-service-3 Created 0.1s
✓Container challenge3-node-service-2 Created 0.1s
✓Container challenge3-nginx-1 Created 0.0s
Attaching to db-1, nginx-1, node-service-1, node-service-2, node-service-3
db-1 | 2024-04-21 18:42:51+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:11.3.2+maria~ubu2204 started.
node-service-3 | Server running on port 3000
node-service-2 | Server running on port 3000
db-1 | 2024-04-21 18:42:52+00:00 [Warn] [Entrypoint]: /sys/fs/cgroup/name=systemd:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 14:misc:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 13:rdma:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 12:pids:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 11:hugetlb:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 10:net_prio:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 9:perf_event:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 8:net_cls:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 7:freezer:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 6:devices:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 5:memory:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 4:blkio:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 3:cpuacct:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 2:cpu:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 1:cpuset:/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822
db-1 | 0::/docker/a49d8a215520d3f752a10edcf31e3c98a205a519bf0cca4caa88889eaa72b822/memory.pressure not writable, functionality unavailable to MariaDB
db-1 | 2024-04-21 18:42:52+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
db-1 | 2024-04-21 18:42:52+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 1:11.3.2+maria~ubu2204 started.
node-service-1 | Server running on port 3000
db-1 | 2024-04-21 18:42:52+00:00 [Note] [Entrypoint]: MariaDB upgrade not required
db-1 | 2024-04-21 18:42:53 0 [Warning] Setting lower_case_table_names=2 because file system for /var/lib/mysql/ is case insensitive
db-1 | 2024-04-21 18:42:53 0 [Note] Starting MariaDB 11.3.2-MariaDB-1:11.3.2+maria~ubu2204 source revision 068a6819eb63bcb01fdfa037c9bf3bf63c33ee42 as proces
s 1
nginx-1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
nginx-1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx-1 | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
nginx-1 | 10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
nginx-1 | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
nginx-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh


File Edit Selection View Go Run Terminal Help
docker assignment
EXPLORER
DOCKER ASSIGNMENT
challenge1
challenge2
challengefinal
db
db-data
docker
api
Dockerfile
package.json
server.js
db
init
init.sql
Dockerfile
nginx
Dockerfile
nginx.conf
.env
docker-compo...
images
Docker Assignment.pdf
README.md
student.cfg
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
TERMINAL
docker-compose
nginx-1 | 172.30.0.1 - - [24/Apr/2024:21:40:22 +0000] "GET /api/books HTTP/1.1" 500 26 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/124.0.0.0 Safari/537.36" "-"
node-service-2 | at Protocol._parsePacket (/app/node_modules/mysql/lib/protocol/Protocol.js:291:23)
node-service-2 | at Parser._parsePacket (/app/node_modules/mysql/lib/protocol/Parser.js:433:10)
node-service-3 | {
node-service-3 | status: 'success',
node-service-3 | contents: { MemFree: 4584136, MemAvailable: 6343960 },
node-service-3 | pid: 1,
node-service-3 | hostname: '14136de17f63',
node-service-3 | counter: 0
node-service-3 | }
nginx-1 | 172.30.0.1 - - [24/Apr/2024:21:40:27 +0000] "GET /api/stats HTTP/1.1" 200 120 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
, like Gecko) Chrome/124.0.0.0 Safari/537.36" "-"
node-service-1 | {
node-service-1 | status: 'success',
node-service-1 | contents: { MemFree: 4583884, MemAvailable: 6343708 },
node-service-1 | pid: 1,
node-service-1 | hostname: 'a425e4d35a85',
node-service-1 | counter: 0
node-service-1 | }
nginx-1 | 172.30.0.1 - - [24/Apr/2024:21:40:30 +0000] "GET /api/stats HTTP/1.1" 200 120 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
, like Gecko) Chrome/124.0.0.0 Safari/537.36" "-"
node-service-2 | {
node-service-2 | status: 'success',
node-service-2 | contents: { MemFree: 4583632, MemAvailable: 6343456 },
node-service-2 | pid: 1,
nginx-1 | 172.30.0.1 - - [24/Apr/2024:21:40:31 +0000] "GET /api/stats HTTP/1.1" 200 120 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
, like Gecko) Chrome/124.0.0.0 Safari/537.36" "-"
node-service-2 | hostname: 'ac07e17fdff3',
node-service-2 | counter: 0
node-service-2 | }
node-service-3 | {
node-service-3 | status: 'success',
node-service-3 | contents: { MemFree: 4585688, MemAvailable: 6345520 },
nginx-1 | 172.30.0.1 - - [24/Apr/2024:21:40:31 +0000] "GET /api/stats HTTP/1.1" 200 120 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
, like Gecko) Chrome/124.0.0.0 Safari/537.36" "-"
node-service-3 | pid: 1,
node-service-3 | hostname: '14136de17f63',
node-service-3 | counter: 0
node-service-3 | }
node-service-1 | {
node-service-1 | status: 'success',
main*
Ln 20, Col 27 Spaces: 2 UTF-8 CRLF Compose Go Live Prettier


localhost:8888/api/stats
Pretty-print
{
"status": "success",
"contents": {
"MemFree": 5508948,
"MemAvailable": 6362260
},
"pid": 1,
"hostname": "f8764dadb5e6",
"counter": 0
}


localhost:8888/api/stats
Pretty-print
{
"status": "success",
"contents": {
"MemFree": 5505204,
"MemAvailable": 6360388
},
"pid": 1,
"hostname": "a6567540e105",
"counter": 0
}


localhost:8888/api/stats
Pretty-print
{
"status": "success",
"contents": {
"MemFree": 5506264,
"MemAvailable": 6361452
},
"pid": 1,
"hostname": "56e67245cd05",
"counter": 0
}