TECHNICAL UNIVERSITY OF KOSICE

Faculty of Electrical Engineering and Informatics

CLOUD TECHNOLOGIES

Assignment-1: Docker

DONE BY:

Mohammed Niaz Khaleel Jameel

TABLE OF CONTENTS

Conditions for deploying and launching the application.

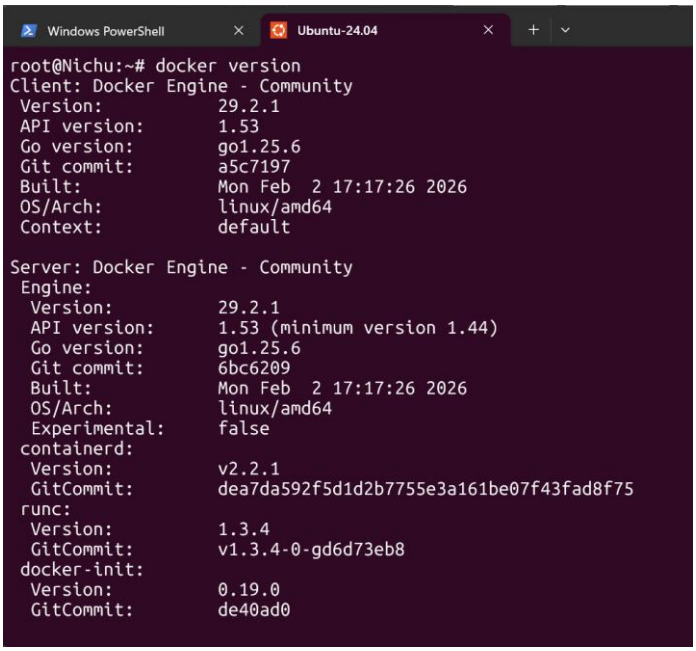
Required software.

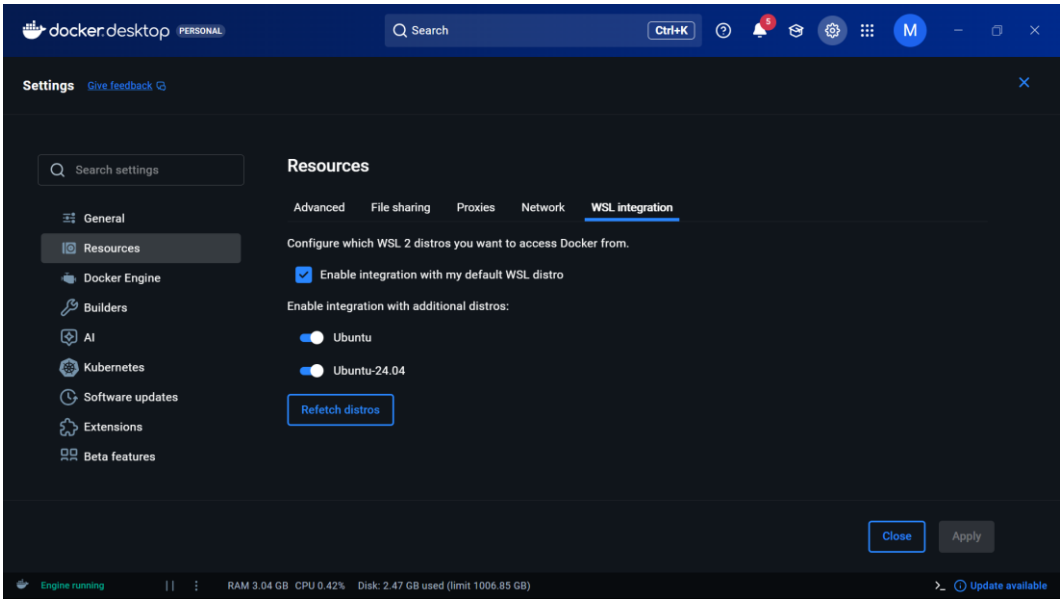
1. Operating System

- Windows 10/11 with WSL2 (Windows Subsystem for Linux) enabled.

2. Docker Engine

- Version 20.10 or newer
- Command to check whether it is installed: **docker --version.**
- On Windows we need Docker Desktop with WSL2 backend enabled





3. Docker Compose

- Version 2.0 or maybe any newer version will be fine.
- Command to check its usage: **docker compose version**

```
root@Nichu:~# docker compose version
Docker Compose version v2.40.2-desktop.1
root@Nichu:~#
```

4. Internet connection

- Required only during prepare-app.sh to pull these images from Docker Hub:
    - postgres:15-alpine
    - nginx:alpine
    - python:3.11-slim

      These were used to build the app image
- Since python, postgresSQL, nginx runs inside the container and Django installed inside the container via pip, we don’t need to install these manually

Hardware requirements

- At least **1 GB free RAM** for all three containers
- At least **2 GB free disk space** for Docker images

Network requirements

- Port **80** must be free on the host machine. Make sure it is not used by another web server.

Description of the application

My Diary is a private, multi-user web diary application accessible through my localhost. It allows multiple users to each have their own personal diary, separate from other users.

Things an user can do:

❖ **Account management**

1. Register a new personal account with a username and password.
2. Log in and log out securely.
3. Each user can only see their own entries — never anyone else's.

❖ **Writing diary entries**

1. Create a new diary entry with a title, written content, and a mood tag.
2. Edit an existing entry at any time.
3. Delete an entry permanently.

❖ **Browsing entries**

1. View all past entries displayed as cards on the home page, sorted newest first
2. Click any entry card to read the full entry
3. Search entries by keyword — searches both the title and the content
4. Filter entries by mood

❖ **Data persistence**

1. All diary entries are stored in a PostgreSQL database. The data is saved in a Docker named volume (postgres_data) which means entries survive container restarts, application updates, and even stopping and restarting the entire application.

❖ **Admin panel**

1. A Django admin panel is available at /admin/ for superusers to manage all users and entries directly.

Virtual networks and named volumes.

❖ The Virtual networks here is: **diary_network**

❖ A private Docker bridge network that connects all three containers together.

❖ Containers communicate with each other using container names as hostnames instead of IP addresses**.**

❖ **diary-nginx** talks to **diary-app** using the hostname app on port 8000, and **diary-app** talks to diary-db using the hostname db on port 5432.

❖ From outside, only diary-nginx is reachable on port 80 where the database and Django app are completely hidden from the internet.

- Without this network, containers cannot see each other at all.
- Some of the named volume used are: **postgres_data** and **diary_static**
- **Postgres_data** is mounted inside the **diary-db** container at /var/lib/postgresql/data.
- Stores the entire PostgreSQL database of all user accounts and all diary entries.
- This volume lives independently of the containers, so when you stop the application the data stays on disk and is available again when you restart.
- It is only permanently deleted when **running remove-app.sh.**
- Likewise, **diary_static** is mounted inside **diary-app** at /app/staticfiles with write access, and inside **diary-nginx** at the same path as read-only.
- When **diary-app** starts, Django writes all static files (CSS stylesheets) into this volume.
- Nginx then reads directly from the same volume to serve them to the browser, without going through Django at all.
- This makes static file serving faster and more efficient.

Container configuration

- The **diary-db** container runs postgres:15-alpine and is configured entirely through environment variables, POSTGRES_DB, POSTGRES_USER, and POSTGRES_PASSWORD, which set up the database name and credentials on first startup. It mounts the

postgres_data named volume to persist all data and is not exposed on any host port, making it only reachable from within **diary_network.**

- The **diary-app** container is built from a custom docker file based on python:3.11-slim. It installs all Python dependencies from requirements.txt and runs Django via gunicorn with 2 worker processes. A custom entrypoint.sh script runs on startup and it waits until PostgreSQL is ready, runs database migrations automatically, then starts Gunicorn. The container is configured through environment variables for the database connection, secret key, and debug mode. It mounts the **diary_static** volume to share static files with Nginx.
- The **diary-nginx** container runs **nginx:alpine** and is the only container exposed to the host on port 80. Its configuration file nginx.conf is attached into the container, telling Nginx to serve static files directly from the **diary_static** volume and forward all other requests to diary-app on port 8000.

List of Containers

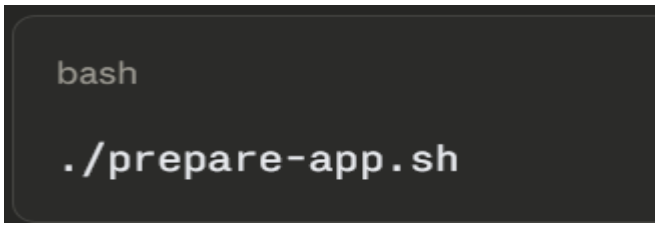
There are three containers used for running this web application.

1. **diary_db –** Image used is postgres: 15-alpine which is a relational database that stores all user accounts and diary entries.

2. **diary_app –** Image used is diary_app which runs on port 8000. It runs Django web app served by gunicorn and this handles all business logic.
3. **diary_nginx –** Image used is nginx which runs on port 80. It is basically a reverse proxy that receives all browser requests,serves static files directly and forwards everything else to Django.

Instructions on how to prepare, launch, pause and delete the application.

❖ To prepare, we will be using the command:



❖ This will build the docker image and creates the network and volumes. This has to be run once before starting the application for the first time or maybe after changing the code.

```
[+] Building 1/1
 ✔diary-app:latest  Built

Preparation complete.
  Image   : diary-app:latest
  Network: diary_network
  Volumes: postgres_data, diary_static

Run ./start-app.sh to start the application.
root@Nichu:~/z1#
```

❖ Next to launch we use:

```bash
./start-app.sh
```

❖ This starts all three containers mentioned previously. The app will be then viewed at localhost.

```
root@Nichu:~/z1# ./start-app.sh
Running app...
[+] Running 3/3
 ✔Container diary-db     Healthy
 ✔Container diary-app    Started
 ✔Container diary-nginx  Running

All containers started.
The app is available at http://localhost

Note: First startup may take 10-15 seconds while the
      database initialises and migrations run.
root@Nichu:~/z1#
```

❖ To pause, we use:

```bash
./stop-app.sh
```

❖ Stops and removes all containers but keeps the postgres_data and diary_static volumes just there. All diary entries are preserved. The application can be seen afterwards after launching it.

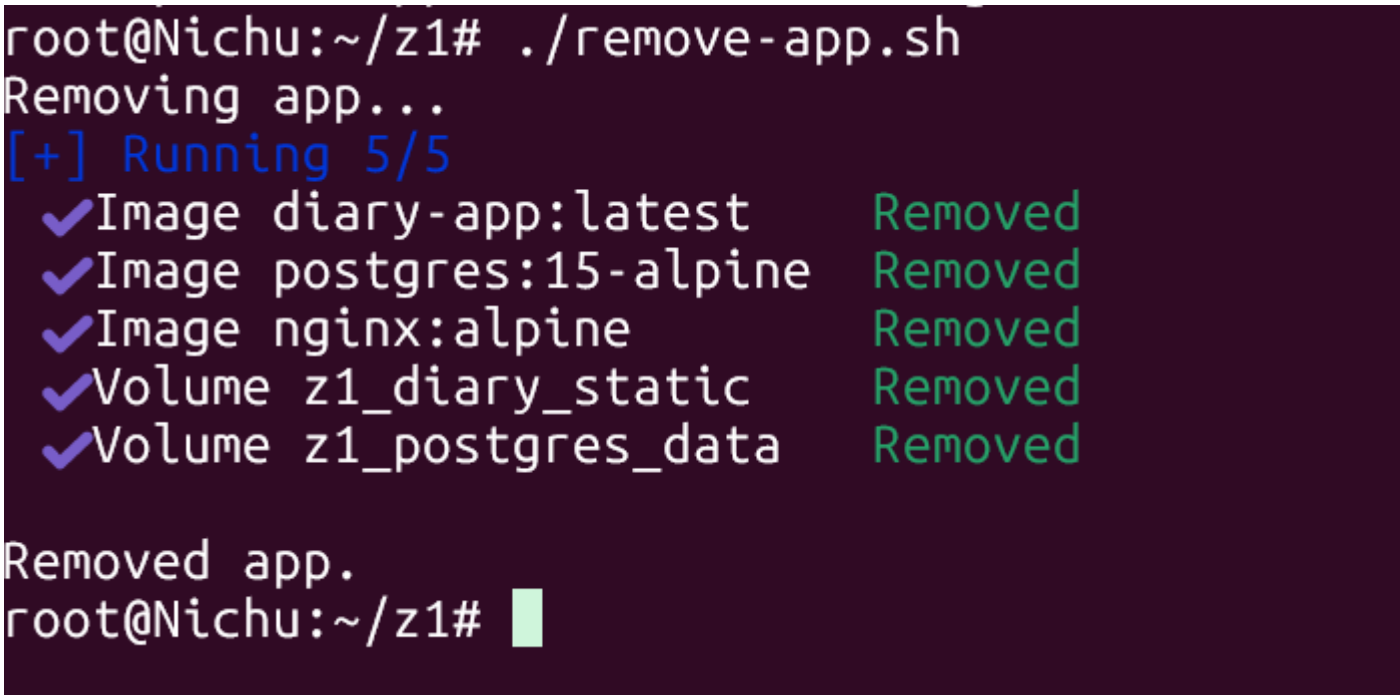
```
root@Nichu:~/z1# ./stop-app.sh
Stopping app...
[+] Running 4/4
 ✔Container diary-nginx      Removed
 ✔Container diary-app        Removed
 ✔Container diary-db         Removed
 ✔Network z1_diary_network   Removed

App stopped. All data is preserved in the pos
Run ./start-app.sh to start it again.
root@Nichu:~/z1#
```

❖ To delete that container, we use:

```bash
./remove-app.sh
```

❖ This deletes the entire containers, images, etc., permanently. You may need to again initialize by creating it first.



How to view the application in a web browser?

❖ To view the application, once our application is prepared, we need to launch it using the **./start-app.sh** .

❖ After few seconds, the link to local host will be created and we can use any of our desktop's web browser to view.

```
root@Nichu:~/z1# ./start-app.sh
Running app...
[+] Running 3/3
 ✔Container diary-db     Healthy
 ✔Container diary-app    Started
 ✔Container diary-nginx  Running

All containers started.
The app is available at http://localhost

Note: First startup may take 10-15 seconds while th
      database initialises and migrations run.
root@Nichu:~/z1#
```

My Diary
Login Register

Welcome back

Username
Password
Login

No account? Register here

List of used resources

- Docker documentation
- Django documentation
- Docker Hub

Use of artificial intelligence

I used Claude (claude.ai) by Anthropic as an assistant for guiding me for this project. It helped me with generating code, explaining docker concepts, and fixing configuration issues. All final decisions regarding the application design, structure, and technology choices were made by me.

--END OF DOCUMENT--