Challenge Task “Noteivate”

Version: 1.0

Author: Michael Enzler, Fabio Stocker

Deadline: 22.05.2023

Module: Distributed Systems

Study course: Computer Science

Created: 14. April 2023

Last change: 21. May 2023

Content

1 Introduction

About

This is the documentation by Michael Enzler and Fabio Stocker of the challenge task in the module Distributed Systems. In this report, the documentation of the authors' solution is delivered.

Requirements

The system needs the following components:

- Simple Frontend (e.g., HTML, Vue, React, Svelte)
- Load balancer (e.g., traefik, nginx, HAproxy, Caddy)
- Two instances of a service (your choice), and during the challenge task presentation, one instance will be shut off.
- One storage backend

Additionally, the solution must meet the following requirements:

- Load balancing with scalable service
- Failover of a service instance
- Dockerized
- Frontend connects via HTTP/3
- Persistent storage (storage does not need to be scalable, but you can build it scalable if you want)
- Use latest stable releases of chosen libraries and frameworks.

Summary

The idea to solve this challenge task is to create a notes app that is accessible via a web browser and stores the written notes in a database.

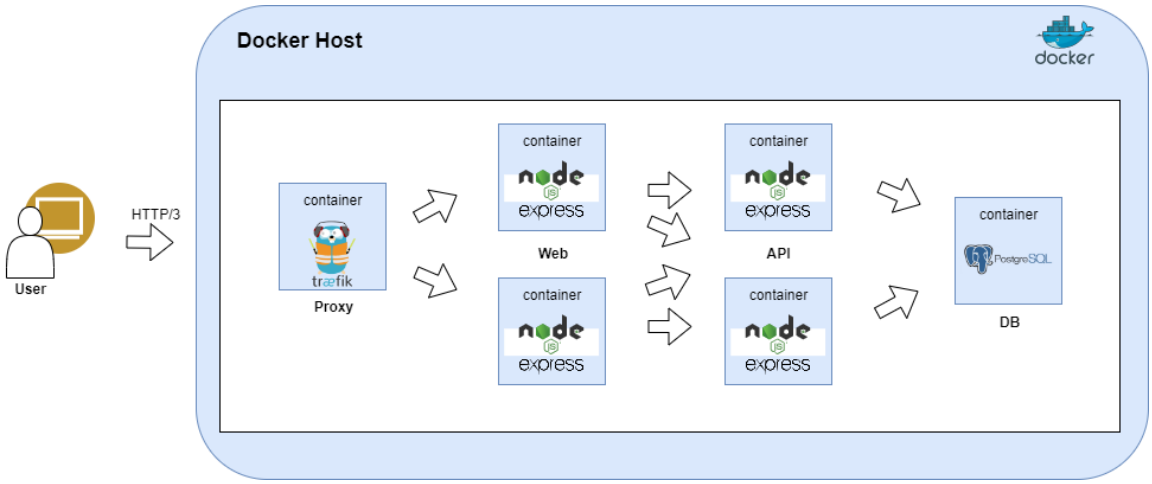
2 Infrastructure

Overview

The following technologies are used for the infrastructure:

- Docker
- Node.js
- Express.js
- PostgreSQL
- Traefik

Architecture



The Noteivate app will be accessed by the user via a web browser. A HTTP/3 connection to the traefik load balancer will be established. The traffic will then be load balanced to one of the two frontend (web) containers. Next, the frontend containers communicate with one of the two API's, which in turn will send a request to the PostgreSQL database. All services have their own container.

Docker

In the Noteivate application, Docker is used to containerize and manage the deployment of the different services, such as the database, API, and web services, as well as the Traefik reverse proxy. Docker Compose is used to define and run the multi-container application, making it easier to set up and start the application with a single command. The docker-compose.yml file contains the configuration of each service and their dependencies. A Docker volume named "noteivate" is created to persist the PostgreSQL database data across container restarts. This approach simplifies the setup process and ensures consistency across different environments.

Proxy

The application makes use of Traefik, a dynamic reverse proxy. Traefik acts as an intermediate for client requests, routing them to appropriate backend services, in this case the web service. It manages SSL termination at the same time, ensuring secure data delivery. SSL certificates are manually generated using OpenSSL. Users can access the program securely by default via https://localhost/. If a user visits the webpage via HTTP, they are redirected to HTTPS automatically. Furthermore, Traefik also functions as a load balancer, distributing and routing traffic among multiple instances of the web service. Additionally, Traefik makes it possible to keep all backend services using HTTP, minimizing the need to use HTTPS while still having a secure infrastructure.

Web

Noteivate comes with a web service that acts as an interactive user interface for note management tasks. This service, built with well-known web technologies such as Node.js and Express, works in tandem with the API service to allow users to add, modify, and delete notes. Because of the existing Traefik reverse proxy configuration, the web service is easily accessible. The web service primarily serves as a simple front-end for communicating with the API, providing a user-friendly platform for note manipulation.

API

The API service functions as a RESTful API, allowing you to do CRUD (Create, Retrieve, Update, Delete) activities on notes. The API is built using Node.js and Express, and communicates with a PostgreSQL database to manage notes data. Despite its current creation with Node.js and Express, it can be redeveloped in any other programming language as long as the endpoints are retained. This is made possible due to the containers being isolated from each other. The API's primary focus is on allowing communication between web and database containers. It does not store any data itself.

Database

The database service is essential in the Noteivate application for persistently storing note data. This is accomplished through PostgreSQL, a widely used database management system. The Docker volume is used to store data, persisting over container restarts. By connecting directly with the database, this configuration enables the API service to manage note data in a secure and efficient manner.

3 Design Decisions

Docker

Docker offers a compelling solution for containerization of applications, providing portability, scalability, reproducibility, flexibility, faster development cycles and resource efficiency. Moreover, it is a requirement to create a dockerized frontend, which means there aren't any other possibilities to choose from.

Express / Node.js

Here at OST in the web engineering modules, it is recommended to use Node.js and Express. They are commonly used and consistently maintained by other developers.

PostgreSQL

To store the user's notes persistently, PostgreSQL will be used as a database. PostgreSQL is well-known in the team, making it easier to work together.

Traefik

The traefik load balancer was mentioned in the lectures in Distributed Systems of Week 4. It seemed to be a good and well-documented solution for the project. Hence, traefik was used.

GitLab

As a student of OST, most of GitLab's features can be used for free since it's self-hosted. This suits the project's needs perfectly. Moreover, the code is to be shared, versioned and tracked during development. There is no need for a complex setup due to the small spectrum of the project.