minor changes
Lukas Ammann authored 4 days ago

149290fb

Readme.md 2.86 KiB

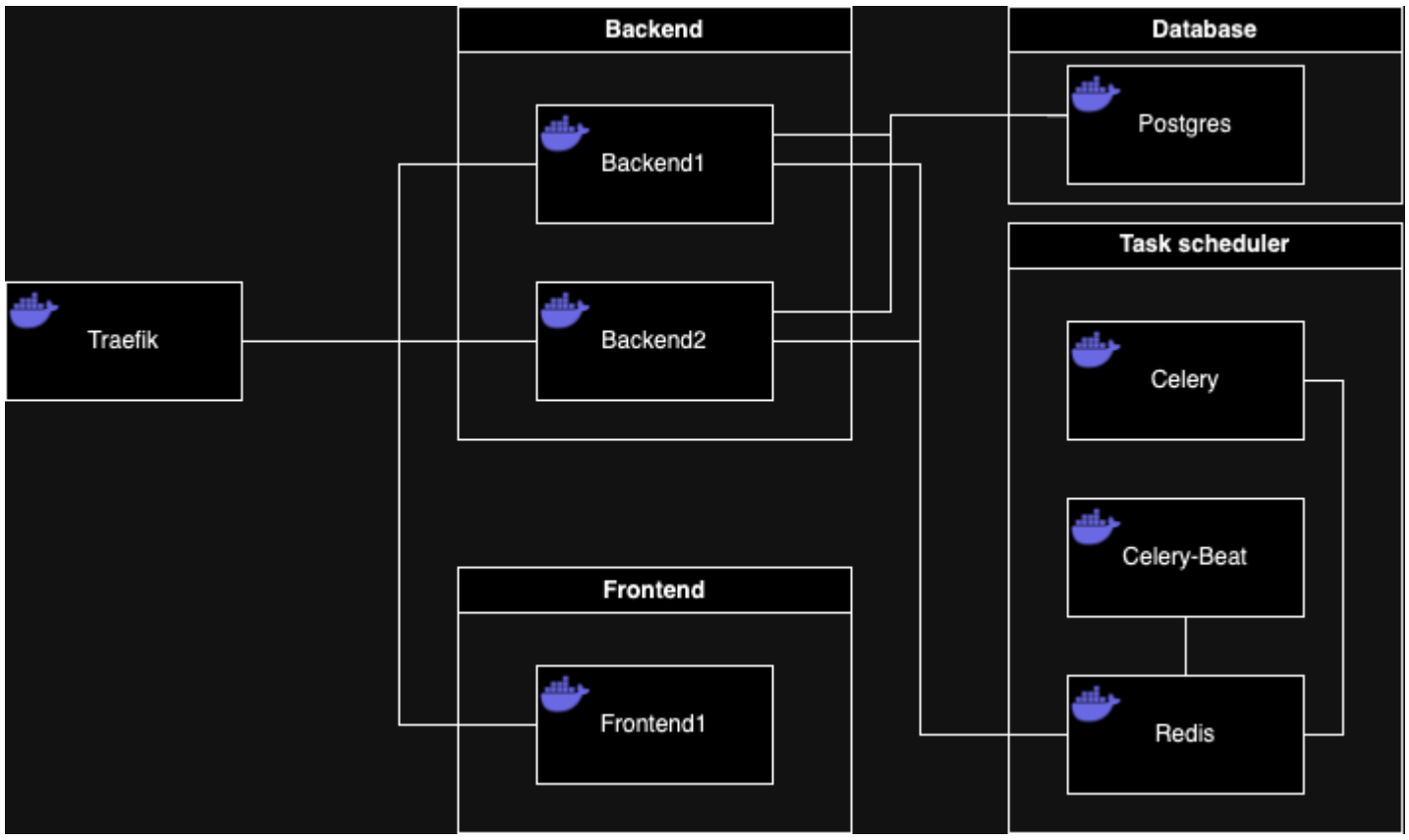
Listify

Your web-shoppinglist

Project documentation of challenge task distributed systems.

Containerization

All services described here are dockerized and defined with docker-compose.



Backend

As the backend we use the django web framework with djangorestframework to implement the api. To ensure stability and high uptime of the system, multiple (2) instances of the backend are running simultaneously. This is achieved by using the "replicas"-option in docker compose.

Database

As a database we use Postgres which gets populated by the backend on startup.

Celery

Celery is used together with celery-beat and redis to perform the distributed task. The distributed task is triggered once per minute and is executed inside one of the workers.
We are aware that this results in some overhead as we need three containers in addition to run celery with all its dependencies. However, we decided to use celery as this is the most widely recommended approach on the internet with our tech stack (django). In addition, the setup of celery is mostly straightforward and easy to use, although we had some starting difficulties.

Distributed Task

The distributed task, as requested in the task description, counts all items inside the database and writes the information into the database. Under the django admin page we can see the gathered statistic (how many products at what time were entered in the system).

Frontend

On the frontend side we use React.JS with Vite. With a React-Router we enable the SPA to have multiple routes. The main route is the

removed. You can also create new items. After creation they have an empty description which can be updated initially. We did not implement any logic to delete any lists or edit any items since that is not the main focus of this challenge task.

Reverse-Proxy

As a reverse-proxy we use traefik. The traefik serves all traffic by default to the frontend container, but if the traffic starts with `/api`, it gets redirected to the backend servers. The load balancing happens by traefik. So even if we turn off a single instance of the backend, the whole application still works. We get an even distribution for the requests to the backend due to round-robin. The registration of the docker containers to traefik is made through docker labels.

Getting Started

Since the node_modules are mounted to the volume to keep them in sync you need to install them locally.

```
cd ./frontend
yarn install
cd ..
docker compose up
```

After everything started, head over to your browser to http://localhost.