Distributed Programming report

LAI Khang Duy - Lylia DJALI

Université de Paris

29-01-2022

Contents

List of Figures

1 Bonus work Qwiklabs

Please find the link here

- LAI KHANG DUY
- LYLIA

2 Resource

Please find the source code of the project here.

CLICK HERE

3 Présentation du projet

The purpose of the project is to prove the concept of using multiple technologies for distributed programming. With this project, we have built a full web server with 2 backends, 1 frontend and 1 database to showcase the problems.

The application has 2 main functionalities. 1. Search and display the weather and the temperature of the place that user type in. 2. When the user click record, the application will write the data to the database, and display them the next time user visit the website.

Each of it is under a Docker container and defined to comunicate with each other using Docker Compose.

The front end fetch from 2 backends with 2 independent tasks, thus we deployed the concept of microservices.

Notre projet consiste à réaliser une application web tout en utilisant les technologies acquises lors du cours Programmation distribuée.

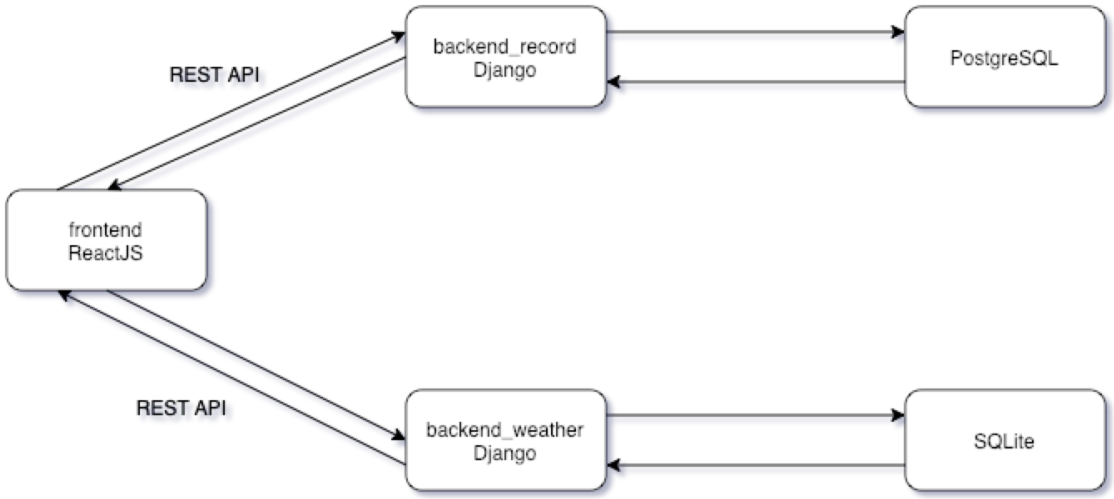

Figure 1: Architecture de l'application

Et pour cela nous avons créé une application qui affiche la température pour n'importe quelle ville au monde ainsi qu'un enregistrement des températures précédentes récupéré depuis la base de données.

4 Introduction about the technology

4.1 Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

4.2 React.JS

ReactJS is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Facebook and a community of individual developers and companies. It is the most common frontend framework for building web application at the moment.

4.3 PostgreSQL

PostgreSQL is a free and open-source relational database management system (RDBMS) emphasizing extensibility and SQL compliance. The reason we choose Postgres over other Database is that it is fully intergrated with Django. Furthermore, the official image of PostgreSQL on Dockerhub is very easy to deploy with out much modification.

5 Architecture de l'application

Pour expliquer le fonctionnement de notre application web, nous avons schématisé l'architecture logicielle de notre application comme on peut le voir sur la figure ci-dessous:

```
->project_final/
  ->backend_record/
  ->backend_weather/
  ->frontend/
  ->.gitignore
  ->docker_compose.yml
```

Notre application web utilisant le web service REST est composé de :

1. Un front end (qui nous sert de navigateur) codé en ReactJS.

This will take the data from backend_weather and backend_record and display them to the screen for the user.

2. Deux back end (qui nous sert de navigateur) codé en Python.

- Un qui sert à stocker et à extraire les données depuis notre base donnée codé en python en utilisant Django (backend_weather).
- Le deuxième qui sert à récupérer la température d'une autre API codé en python en utilisant Django (backend_record).

3. Base de donnée PostgreSQL.

- The backend_record will take the record from the moment user click the record button and write it in PostgreSQL.

The SQLite from backend_weather is running in the same container with backend_weather to manage the admin task only.

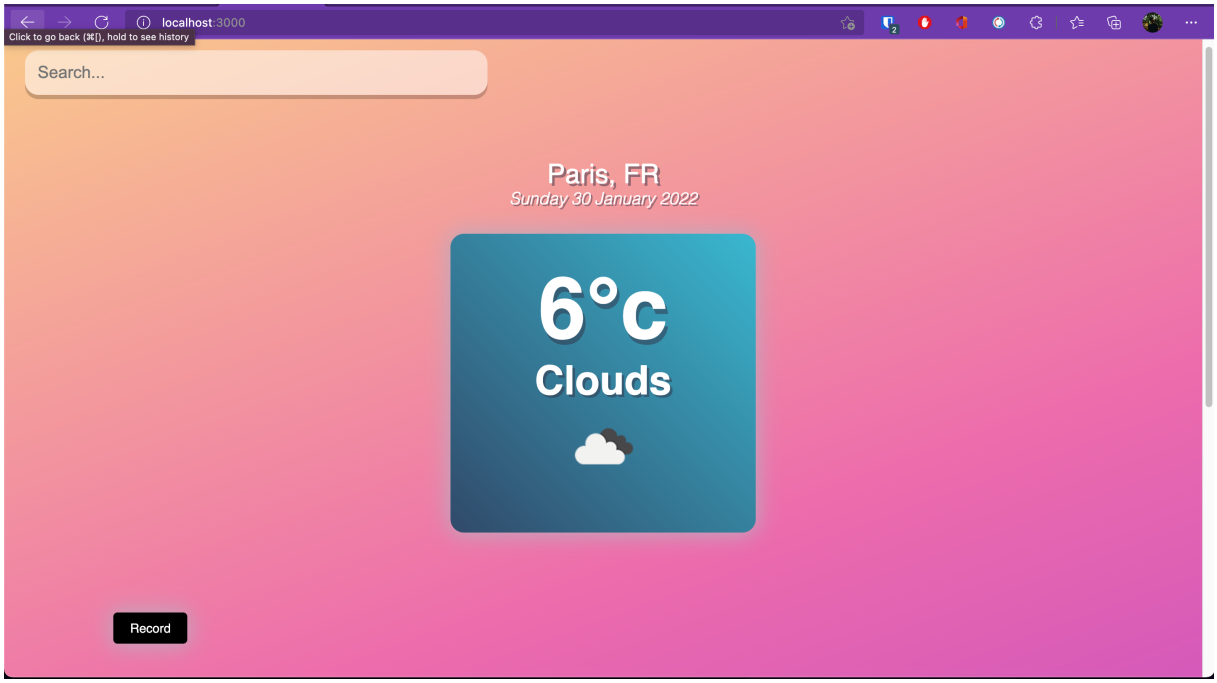

Figure 2: Show weather module

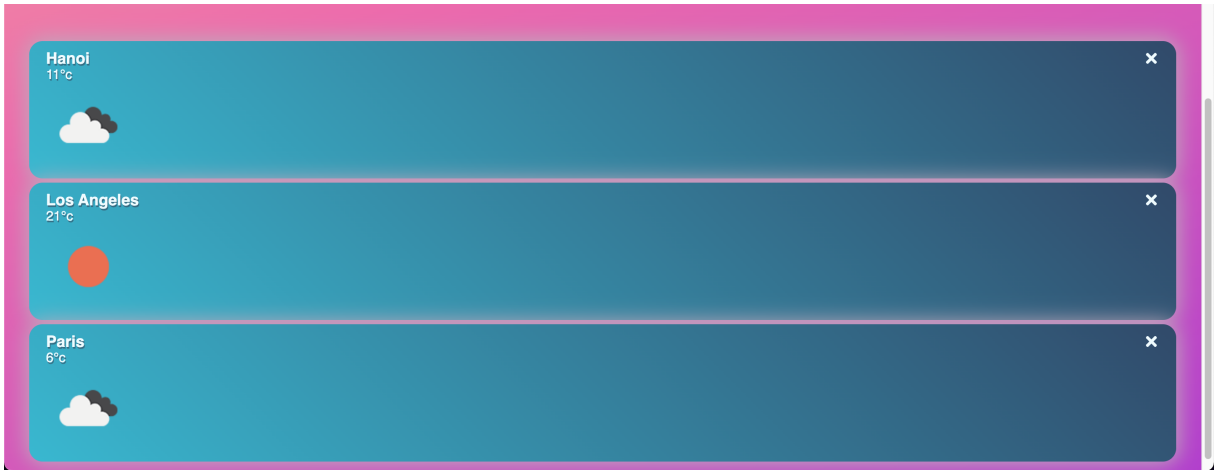

Figure 3: Show record module

6 Distributed programming technologies used in the project

6.1 Docker

Les conteneurs fonctionnent un peu comme les VM, mais de manière beaucoup plus spécifique et granulaire. Ils isolent une seule application et ses dépendances - toutes les bibliothèques logicielles externes dont l'application a besoin pour fonctionner - à la fois du système d'exploitation sous-jacent et des autres conteneurs.

Dans notre application chaque application tourne et contenue dans un docker container, Ce qui permet une utilisation plus efficace des ressources du système, des cycles de livraison de logiciels plus rapides, mais surtout très efficace dans architecture micro-service telle que la nôtre.

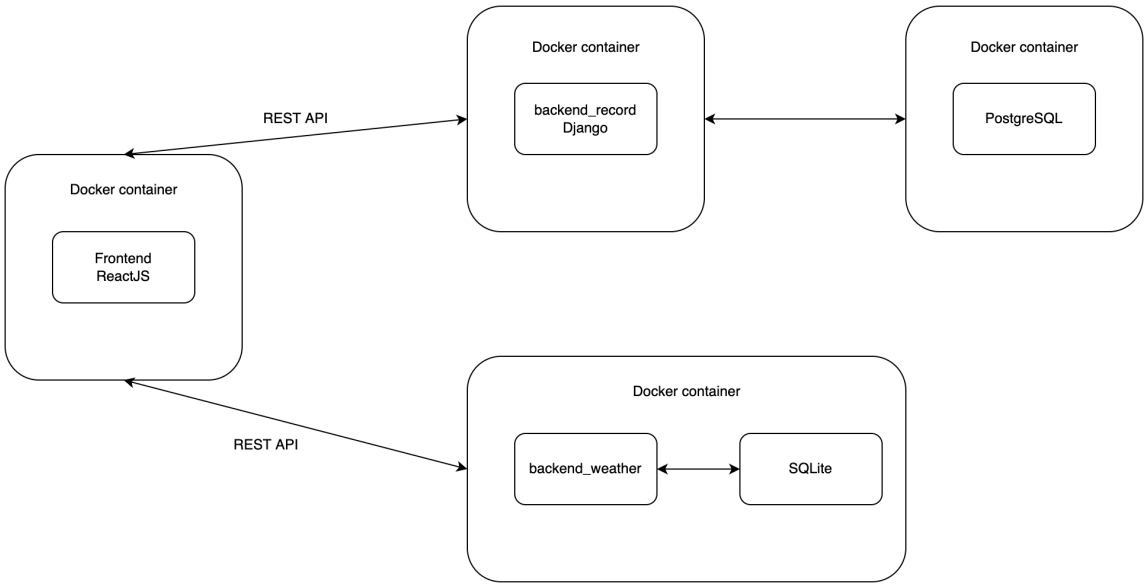

Figure 4: Architechture of Application on docker container

Each of the service defined by a Dockerfile which is the folder repository.

```
FROM python:3
ENV PYTHONUNBUFFERED 1

# COPY . /usr/src/app
WORKDIR /usr/src/app
COPY requirements.txt ./
RUN pip install -r requirements.txt


```

```
COPY . ./
RUN ["python", "manage.py", "makemigrations" ]
EXPOSE 8001
```

Example of the Dockerfile of the Django services

6.2 Docker compose

Compose est un outil permettant de définir et d'exécuter des applications Docker multi-conteneurs. Avec Compose, on utilise un fichier YAML pour configurer les services de votre application. Ensuite, avec une seule commande, on crée et démarre tous les services à partir de notre configuration.

```
version: '3'

services:
  db:
    image: postgres
    ports:
      - "5432:5432"
    environment:
      - POSTGRES_USER=docker
      - POSTGRES_PASSWORD=Docker_123
      - POSTGRES_DB=my_db


  django-backend-record-service:
    build: ./backend_record/
    volumes:
      - ./backend_record:/usr/src/app
    ports:
      - 8001:8001
    command: python manage.py runserver 0.0.0.0:8001
    depends_on:
      - db


  django-record-weather-service:
    build: ./backend_weather/
    volumes:
      - ./backend_weather:/usr/src/app
    ports:
      - 8000:8000
    command: python manage.py runserver 0.0.0.0:8000


  react:
    restart: always
    command : npm start
    build:
```

```
      context: ./frontend/
      dockerfile: Dockerfile
    tty: true
    ports:
      - "3000:3000"
    stdin_open: true
    depends_on:
      - django-backend-record-service
      - django-record-weather-service
```

As you can see, we have totally 4 services defined in the docker compose file

The port of each services is exposed so we can check

6.3 Kubernetes

minikube est un outil qui nous permet d'exécuter Kubernetes localement. minikube exécute un cluster Kubernetes à un seul nœud sur notre ordinateur. Dans notre cas , comme on a utilisé minikube , tous nos conteneurs sont orchestrés par un seul node.

We use Kompose to translate the docker_compose.yml to Kurbernetes resources.

```
kompose convert -f docker-compose.yaml

kubectl apply -f .

kubectl get po
```

7 Conclusion

Ce projet nous a initié aux notions Docker ainsi que Kubernetes d'ou Docker aide à "créer" des conteneurs, et Kubernetes permet de les "gérer" au moment de l'exécution. Utiliser Docker pour emballer et expédier l'application et utiliser Kubernetes pour déployer et mettre à l'échelle l'application. Utilisés ensemble, Docker et Kubernetes servent de facilitateurs de la transformation numérique et d'outils pour une architecture cloud moderne.