

**Auteurs :** Bilal Dembele & Kwameh Dhegbo

**Classe :** X-BAC-DEV-2

## Projet DevOps : O-Kyoto

### 1. Présentation du projet

**O-Kyoto** est une application web de type Single Page Application (SPA) représentant un restaurant japonais fictif situé à Paris. Le projet met en œuvre une architecture **microservices** avec des technologies **Docker**, **Docker Compose**, **Node.js** et **Nginx** pour le frontend. Il respecte les contraintes du sujet imposé : DevOps, CI/CD et conteneurisation.

L'application est entièrement responsive et comporte plusieurs sections : Home, About, Menu, Contact.

### 2. Architecture du projet

Le projet est divisé en 4 conteneurs distincts, chacun étant un microservice :

- **Frontend** : SPA statique servie par Nginx (port 8081)
- **Service Home** : Microservice Node.js pour la section d'accueil (port 3001)
- **Service Menu** : Microservice Node.js pour les plats (port 3002)
- **Service Contact** : Microservice Node.js pour la page contact (port 3003)

Les services sont orchestrés avec **Docker Compose**.

### 3. Technologies utilisées

- Node.js (v18)
- Nginx (alpine)
- Docker / Docker Compose
- TailwindCSS
- GitHub pour le versioning

### 4. Instructions de lancement

1. Cloner le dépôt :

```
git clone https://github.com/Berlal/O-Kyoto.git
```

```
cd O-Kyoto
```

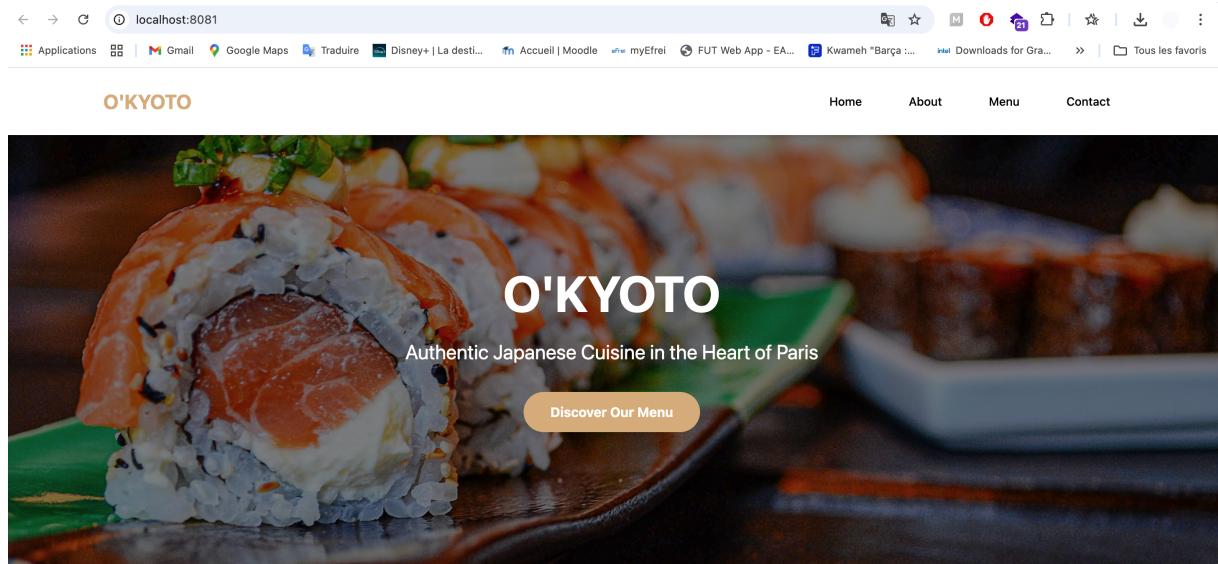
**Auteurs :** Bilal Dembele & Kwameh Dhegbo

**Classe :** X-BAC-DEV-2

2. Lancer tous les services :
3. docker-compose up --build
4. Accéder à l'application sur :  
<http://localhost:8081>

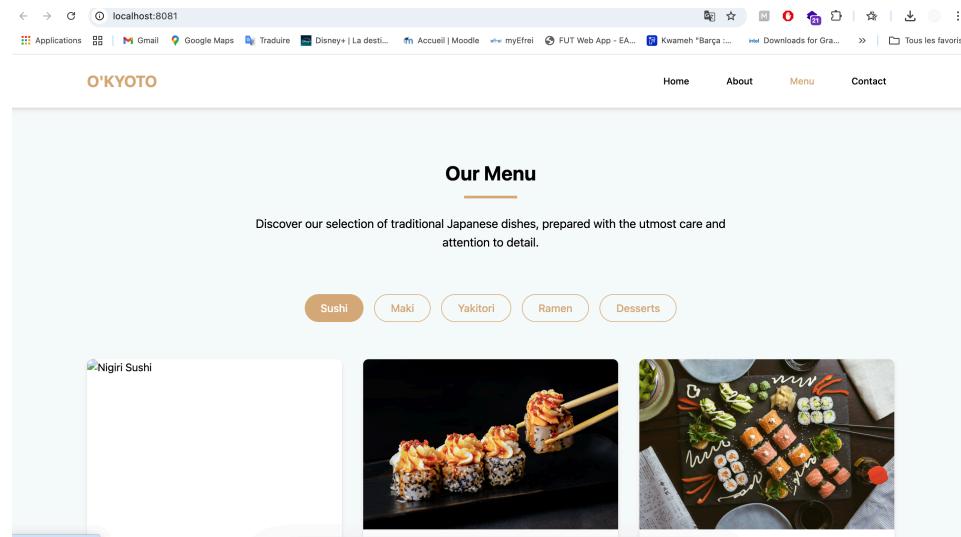
## 6. Captures d'écran

Page d'accueil (<http://localhost:8081>)



### Our Story

Section Menu (avec les différents plats visibles)



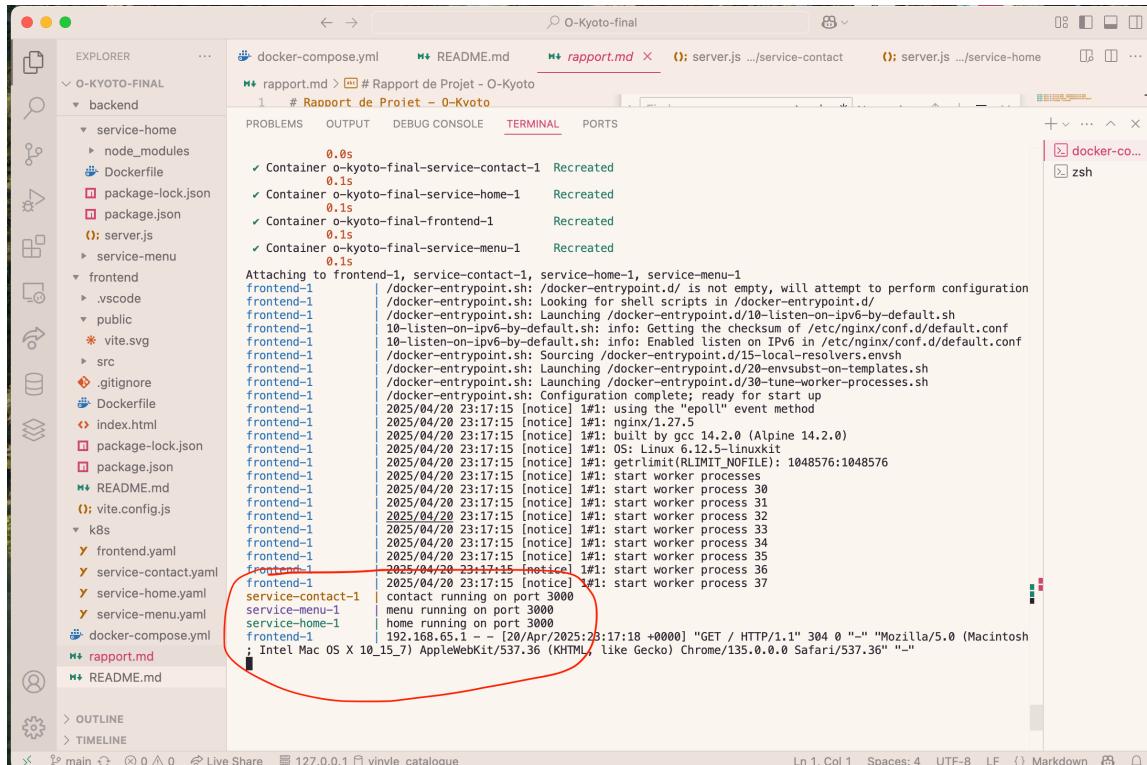
**Auteurs :** Bilal Dembele & Kwameh Dhegbo

## **Classe : X-BAC-DEV-2**

## Docker Compose en cours d'exécution

Container Status						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b6141fcd8691d	o-kyoto-final-frontend	"./docker-entrypoint.s..."	3 minutes ago	Up 3 minutes	0.0.0.0:8081->80/tcp	o-kyoto-final-frontend
-1						
642fe5838941	o-kyoto-final-service-contact	"docker-entrypoint.s..."	3 minutes ago	Up 3 minutes	0.0.0.0:3003->3000/tcp	o-kyoto-final-service-
contact-1						
61db98106950	o-kyoto-final-service-home	"docker-entrypoint.s..."	3 minutes ago	Up 3 minutes	0.0.0.0:3001->3000/tcp	o-kyoto-final-service-
home-1						
celd5d0aa499	o-kyoto-final-service-menu	"docker-entrypoint.s..."	3 minutes ago	Up 3 minutes	0.0.0.0:3002->3000/tcp	o-kyoto-final-service-
menu-1						
df86259a6b7b	vinyle-web-web	"docker-php-entrypoi..."	46 minutes ago	Up 46 minutes (healthy)	0.0.0.0:8080->80/tcp	vinyle-web-web-1
cc3d04052fd4	mysql:8.0	"docker-entrypoint.s..."	46 minutes ago	Up 46 minutes (healthy)	33060/tcp, 0.0.0.0:3307->3306/tcp	vinyle-web-db-1
kwamehleghogeo	mac ~ %					

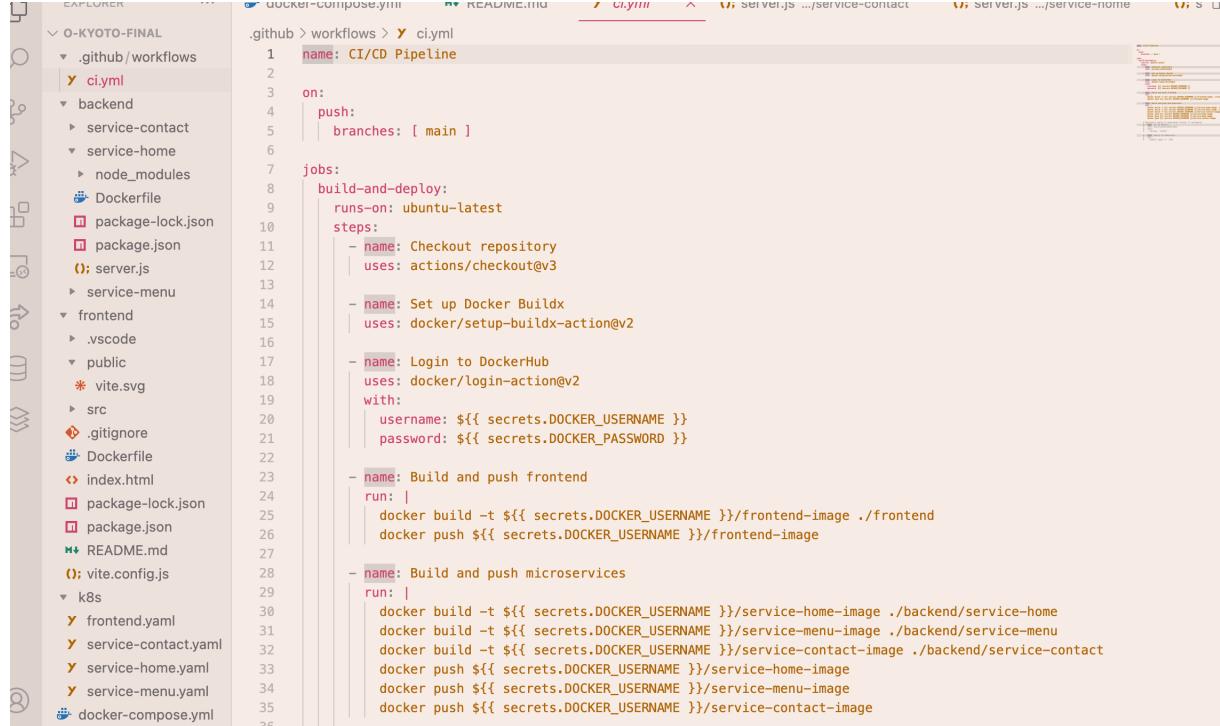
Logs Docker montrant que les services sont lancés



**Auteurs :** Bilal Dembele & Kwameh Dhegbo

**Classe :** X-BAC-DEV-2

Le fichier « .github/workflows/ci.yml » définit un pipeline CI/CD déclenché à chaque push sur la branche main.



```
name: CI/CD Pipeline
on:
  push:
    branches: [ main ]
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v3
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v2
      - name: Login to DockerHub
        uses: docker/login-action@v2
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}
      - name: Build and push frontend
        run:
          docker build -t ${ secrets.DOCKER_USERNAME }/frontend-image ./frontend
          docker push ${ secrets.DOCKER_USERNAME }/frontend-image
      - name: Build and push microservices
        run:
          docker build -t ${ secrets.DOCKER_USERNAME }/service-home-image ./backend/service-home
          docker build -t ${ secrets.DOCKER_USERNAME }/service-menu-image ./backend/service-menu
          docker build -t ${ secrets.DOCKER_USERNAME }/service-contact-image ./backend/service-contact
          docker push ${ secrets.DOCKER_USERNAME }/service-home-image
          docker push ${ secrets.DOCKER_USERNAME }/service-menu-image
          docker push ${ secrets.DOCKER_USERNAME }/service-contact-image
```

## 7. Conclusion

Ce projet nous a permis de concevoir une application web complète en suivant une architecture microservices, avec une conteneurisation Docker sur l'ensemble des services. Le frontend, développé en SPA et stylisé avec TailwindCSS, interagit avec des services indépendants pour plus de modularité. Un pipeline CI a été mis en place via GitHub Actions pour automatiser les builds. L'ensemble est structuré pour pouvoir être déployé facilement, y compris sur Kubernetes. Ce travail en binôme nous a permis d'appliquer concrètement les outils et méthodes DevOps.