



# **RAPPORT                      TECHNIQUE DÉPLOIEMENT    D'APPLICATION FULLSTACK AVEC CI/CD**

**ÉLABORÉ PAR : MOHAMED EL FAROUKI  
ANNÉE UNIVERSITAIRE : 2024/2025**

**ZIAD NAJIM**

## 1. Présentation Générale

### Objectif :

Réalisation d'une application fullstack (React + Express + MySQL) avec :

Conteneurisation via Docker

Pipeline CI/CD automatisé avec GitHub Actions

Tests unitaires et d'intégration

### Fonctionnalités clés :

CRUD complet pour la gestion d'utilisateurs

Interface React responsive

API RESTful documentée

## 2. Mise en Place Technique

### 2.1 Backend (Node.js/Express)

#### Architecture :

app.js → Routes → Contrôleurs → Modèles MySQL

#### Endpoints implémentés :

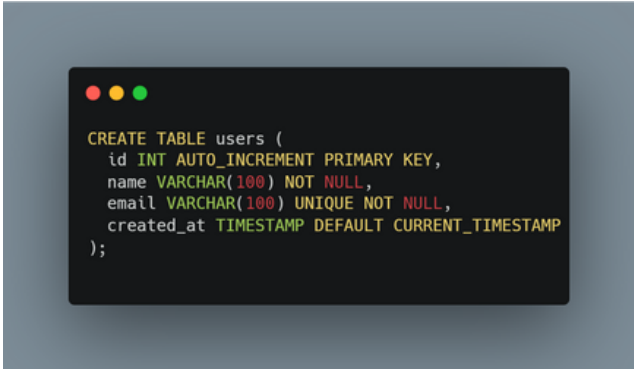
- **GET /api/users** : Récupère la liste complète des utilisateurs
- **POST /api/users** : Crée un nouvel utilisateur
- **PUT /api/users/:id** : Met à jour les informations d'un utilisateur existant (identifié par son ID)
- **DELETE /api/users/:id** : Supprime un utilisateur (identifié par son ID)

### 2.2 Frontend (React)

#### Structure :

- Components : UserList, UserForm
- Services : api.js (Axios)
- Gestion d'état : Context API

## 3. Base de Données MySQL



```
CREATE TABLE users (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  name VARCHAR(100) NOT NULL,  
  email VARCHAR(100) UNIQUE NOT NULL,  
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

## 4. Dockerisation

### 4.1 Configuration

Fichiers clés :

- Dockerfile (Backend) :

```
# Stage 1: Build
FROM node:18-alpine AS builder

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

# Stage 2: Run
FROM node:18-alpine

WORKDIR /app

COPY --from=builder /app/node_modules ./node_modules
COPY --from=builder /app .

ENV PORT=5000
EXPOSE 5000

CMD ["npm", "run", "dev"]
```

docker-compose.yml :

```
version: '3.8'

services:
  mysql:
    image: mysql:8.0
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: usersdb
    volumes:
      - mysql_data:/var/lib/mysql
      - ./mysql/init.sql:/docker-entrypoint-initdb.d/init.sql
    ports:
      - "3306:3306"
    networks:
      - app-network

  backend:
    build: ./backend
    # ... (rest of backend config)
    networks:
      - app-network

  frontend:
    build: ./frontend
    # ... (rest of frontend config)
    networks:
      - app-network

volumes:
  mysql_data:

# THIS NETWORK DEFINITION MUST EXIST
networks:
  app-network:
    driver: bridge
```

## 4.2 Optimisations

Images multi-stage pour réduire la taille

Volumes pour la persistance des données

## 5. Intégration Continue avec GitHub Actions:

- **Workflow CI/CD**

### Étapes principales :

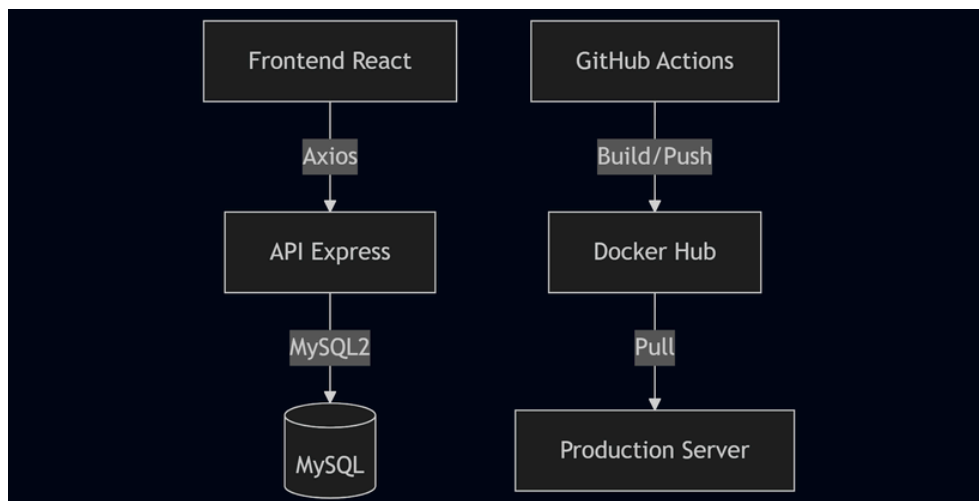
Déclenchement sur push vers main

Exécution des tests unitaires

Build des images Docker

Push vers Docker Hub

- **Schéma d'Architecture**



## Extrait Concis du Workflow CI/CD

```
name: CI/CD Pipeline
on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest
    services:
      mysql:
        image: mysql:8.0
        env: { MYSQL_ROOT_PASSWORD: root, MYSQL_DATABASE: usersdb_test }

    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v3
        with: { node-version: 22 }
      - run: |
          cd backend
          npm ci
          npm test

  deploy:
    needs: test
    steps:
      - uses: docker/login-action@v2
        with: {
          username: ${ secrets.DOCKER_HUB_USERNAME },
          password: ${ secrets.DOCKER_HUB_TOKEN }
        }
      - run: |
          docker compose -f docker-compose.prod.yml build
          docker compose -f docker-compose.prod.yml push
```

### Points Clés :

#### *Déclenchement :*

Sur push/PR vers main

#### *Tests :*

MySQL conteneurisé pour les tests

Installation précise des dépendances (npm ci)

#### *Déploiement :*

Build des images Docker (fichier prod)

Push vers Docker Hub après tests réussis

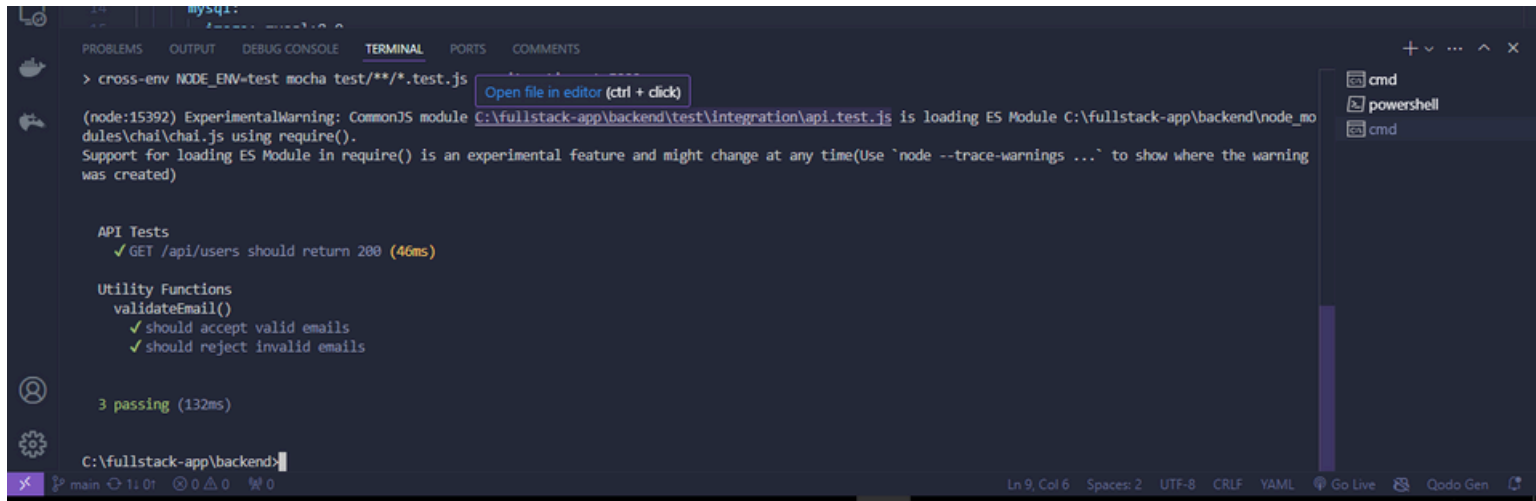
#### *Sécurité :*

Utilisation de secrets pour Docker Hub

Isolation des environnements

## 6. Captures d'Écran

### 6.1 Tests Automatisés



The screenshot shows a VS Code terminal window with the following content:

```
> cross-env NODE_ENV=test mocha test/**/*.test.js
(node:15392) ExperimentalWarning: CommonJS module C:\fullstack-app\backend\test\integration\api.test.js is loading ES Module C:\fullstack-app\backend\node_modules\chai\chai.js using require().
Support for loading ES Module in require() is an experimental feature and might change at any time(Use `node --trace-warnings ...` to show where the warning was created)

API Tests
  ✓ GET /api/users should return 200 (46ms)

Utility Functions
  validateEmail()
    ✓ should accept valid emails
    ✓ should reject invalid emails

3 passing (132ms)
```

The terminal output shows the execution of automated tests using Mocha. The tests are grouped into "API Tests" and "Utility Functions". The "API Tests" section shows a single test passing: "GET /api/users should return 200 (46ms)". The "Utility Functions" section shows two tests passing: "should accept valid emails" and "should reject invalid emails". The overall result is "3 passing (132ms)".

### 6.2 Conteneurs Docker



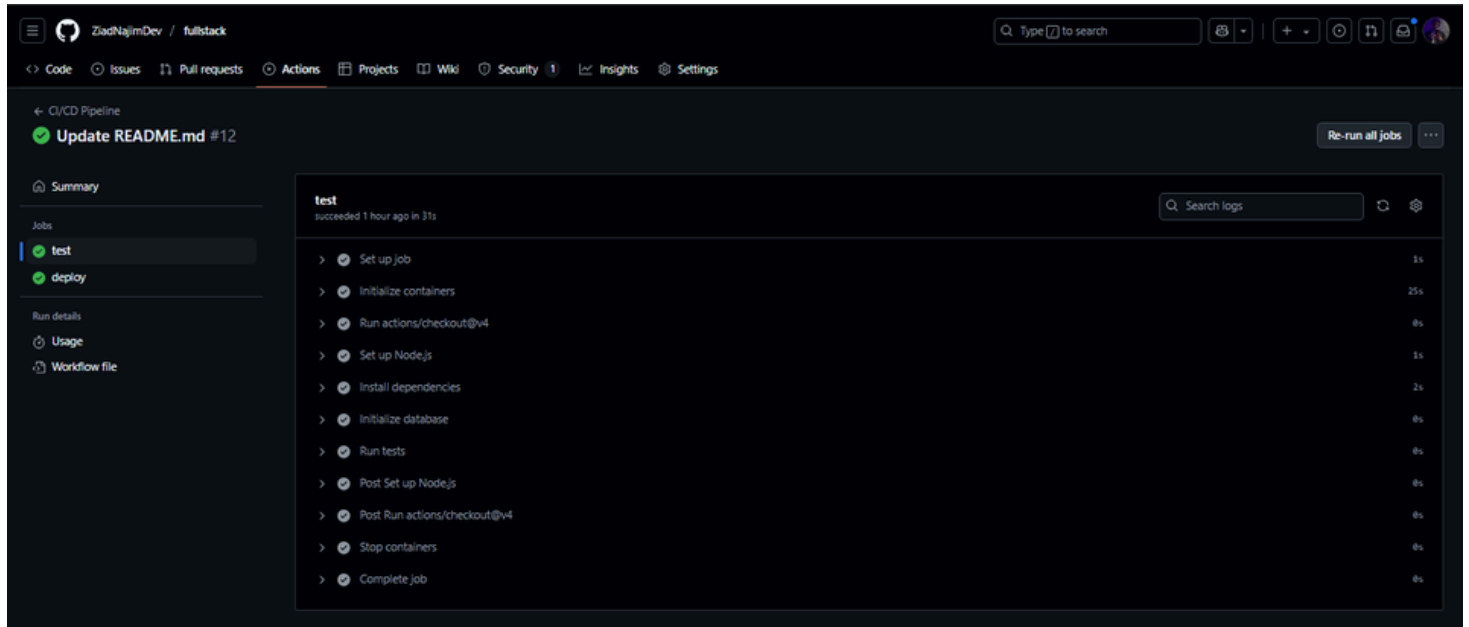
The screenshot shows a VS Code terminal window with the following content:

```
✓ Network fullstack-app_app-network Removed 0.9s
PS C:\fullstack-app> docker volume prune -f
Total reclaimed space: 0B
PS C:\fullstack-app> docker-compose up -d
time="2025-04-10T00:25:17+01:00" level=warning msg="C:\fullstack-app\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 4/4
  ✓ Network fullstack-app_app-network Created 0.2s
  ✓ Container mysql Started 2.2s
  ✓ Container fullstack-app-backend-1 Started 2.1s
  ✓ Container fullstack-app-frontend-1 Started 2.2s
PS C:\fullstack-app> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
b22ead2b78ff	fullstack-app-frontend	"/docker-entrypoint.s..."	38 seconds ago	Up 37 seconds	80/tcp	fullstack-app-frontend-1
28f5567cb4b4	mysql:8.0	"docker-entrypoint.s..."	38 seconds ago	Up 37 seconds	0.0.0.0:3306->3306/tcp, 33060/tcp	mysql
ac36380ecf54	fullstack-app-backend	"docker-entrypoint.s..."	38 seconds ago	Up 37 seconds	5000/tcp	fullstack-app-backend-1

The terminal output shows the execution of Docker commands. The first command is "docker volume prune -f", which removes unused volumes. The second command is "docker-compose up -d", which starts the containers defined in the docker-compose.yml file. The output shows that the network "fullstack-app\_app-network" was removed and then recreated. The containers "mysql", "fullstack-app-backend-1", and "fullstack-app-frontend-1" were started successfully. The "docker ps" command shows the status of the containers, including their IDs, images, commands, creation times, status, ports, and names.

## 6.3 GitHub Actions



## 8. Conclusion et Perspectives

### Bilan :

Application conforme aux spécifications

Pipeline CI/CD fonctionnel

Améliorations possibles :

Ajout de tests End-to-End (Cypress)

Déploiement automatique sur VPS

Monitoring avec Prometheus/Grafana

# ZiadNajimDev/ fullstack



1

Contributor



0

Issues



0

Stars



0

Forks



## ZiadNajimDev/fullstack

Contribute to ZiadNajimDev/fullstack development by creating an account on GitHub.



GitHub

<https://github.com/ZiadNajimDev/fullstack>