

2025

# **DÉVELOPPEMENT FULLSTACK + CI/CD AVEC GITHUB ACTIONS**



**Encadré par : El farouki**

**Réalisé par : darghali noura**



# SOMMAIRE

- 01** Présentation générale du projet
- 02** Étapes de mise en place du backend et frontend
- 03** Explication de la base de données
- 04** Dockerisation : étapes et choix faits
- 05** GitHub Actions : pipeline expliqué étape par étape
- 06** Captures d'écran des tests, actions GitHub, conteneurs Docker
- 07** Difficultés rencontrées et solutions
- 08** Conclusion et axes d'amélioration

# PRÉSENTATION GÉNÉRALE DU PROJET

Le projet docker-fullstack-app est une application fullstack utilisant React.js pour le frontend, Express.js pour le backend et une base de données MySQL. L'application est entièrement dockerisée avec Docker Compose pour faciliter son déploiement et son exécution en environnement isolé. De plus, un pipeline CI/CD a été mis en place avec GitHub Actions pour automatiser le processus de tests, construction et déploiement.

## FONCTIONNALITÉS PRINCIPALES :

- Gestion des utilisateurs (CRUD) via une API REST.
- Interaction avec une base de données MySQL pour stocker les utilisateurs.
- Tests automatisés pour vérifier le bon fonctionnement des fonctionnalités.
- Dockerisation complète pour une portabilité facile et un déploiement simplifié.
- Automatisation avec GitHub Actions pour les tests et le déploiement continu.

# ÉTAPES DE MISE EN PLACE DU BACKEND ET FRONTEND

## Backend :

Le backend a été développé en utilisant Express.js

Voici les étapes principales :

- **Création de l'API** : Les routes pour gérer les utilisateurs (ajout, modification, suppression, récupération) ont été définies dans le fichier `server.js`.
- **Connexion à la base de données** : La connexion à la base de données MySQL est gérée par le fichier `db.js`. Les variables d'environnement nécessaires à la connexion sont stockées dans un fichier `.env`.
- **Gestion des requêtes** : Des requêtes SQL sont exécutées pour gérer les utilisateurs, en utilisant le module `mysql2`.

## Frontend :

Le frontend est développé avec React.js, permettant une interface interactive et dynamique pour gérer les utilisateurs.

- **Création de l'interface** : Les pages ont été créées avec React, en utilisant Axios pour faire les requêtes HTTP vers le backend.
  - **Liste des utilisateurs** : Affiche les utilisateurs récupérés depuis l'API. Chaque utilisateur a des boutons permettant de le supprimer ou de le modifier.
  - **Formulaire d'ajout** : Permet à l'utilisateur d'ajouter un nouvel utilisateur à la base de données en remplissant un formulaire avec des champs tels que le nom et l'email.
- **Gestion de l'état** : L'application utilise React hooks pour gérer l'état local et afficher les utilisateurs dans une liste.

# EXPLICATION DE LA BASE DE DONNÉES

- Création des Dockerfiles pour le Backend et le Frontend

## 1. Dockerfile pour le Backend (API Node.js) :

Le backend est une application Node.js, et son Dockerfile est divisé en deux étapes : la phase de **build** et la phase de **production**.

# Étape 1: Build

FROM node:18-alpine as builder

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY ..

# Étape 2: Production

FROM node:18-alpine

WORKDIR /app

# Copier uniquement les fichiers nécessaires depuis le build

COPY --from=builder /app .

# Ouvrir le port de l'API

EXPOSE 5000

# Lancer le serveur

CMD ["node", "server.js"]

## 2. Dockerfile pour le Frontend (React + Nginx) :

Le frontend est une application React qui est ensuite servie par un serveur Nginx dans un conteneur. Le processus est également divisé en deux étapes :

**Étape 1 : Build avec Node**

**Étape 2 : Serve avec Nginx**

```
# Étape 1: Build avec Node
```

```
FROM node:18-alpine as builder
```

```
WORKDIR /app
```

```
COPY package*.json ./
```

```
RUN npm install
```

```
COPY . .
```

```
RUN npm run build
```

```
# Étape 2: Serve avec nginx
```

```
FROM nginx:alpine
```

```
# Copier les fichiers build React vers nginx
```

```
COPY --from=builder /app/build /usr/share/nginx/html
```

```
# (Facultatif) remplacer la conf nginx
```

```
COPY nginx.conf /etc/nginx/conf.d/default.conf
```

```
EXPOSE 80
```

```
CMD ["nginx", "-g", "daemon off;"]
```

## 2. Fichier docker-compose.yml

Le fichier docker-compose.yml permet de lier tous les services nécessaires à l'application (backend, frontend, base de données MySQL, et phpMyAdmin pour la gestion de la base de données).

### Conteneurs définis dans docker-compose.yml :

- phpmyadmin :
  - Utilisation de l'image phpmyadmin/phpmyadmin pour gérer la base de données MySQL.
  - Il est configuré pour se connecter à la base de données mysql\_db
- backend :
  - Le service backend est construit à partir du Dockerfile
  - Il expose le port 5000 pour permettre l'accès à l'API.
  - Il dépend du service mysql\_db et utilise les variables d'environnement
- frontend :
  - Le service frontend est construit à partir du Dockerfile
  - Il expose le port 3000, mais à l'intérieur du conteneur, Nginx sert l'application sur le port 80.
- mysql\_db :
  - Le service MySQL utilise l'image mysql:8.0.
  - Il expose le port 3306
  - Des volumes sont utilisés pour persister les données de la base de données.

### Définition des Volumes et Réseau :phpmyadmin :

- Volumes :
  - Le volume mysql\_data permet de persister les données MySQL sur la machine hôte,
- Réseau :
  - Les services sont connectés à un réseau privé app-network, ce qui permet la communication entre eux tout en isolant les conteneurs du réseau public.
  - Le driver bridge est utilisé pour créer ce réseau privé.

## 3. Commande pour exécuter les conteneurs Docker

docker-compose up --build

# GITHUB ACTIONS : PIPELINE EXPLIQUÉ ÉTAPE PAR ÉTAPE

Un pipeline CI/CD a été mis en place dans GitHub Actions pour automatiser les tests, la construction des images Docker, et le déploiement de l'application.

Voici les étapes détaillées du processus :

- Création de compte Docker Hub
- création de repository Docker
- Création des secrets GitHub :
  - DOCKER\_USERNAME : Le nom d'utilisateur de compte Docker Hub.
  - DOCKER\_PASSWORD : Le mot de passe de compte Docker Hub.
- Création du fichier .github/workflows/ci.yml avec les étapes:
  - Checkout du code
  - Setup Node.js
  - Installation des dépendances
  - Build de l'image Docker
  - Push de l'image Docker vers Docker Hub

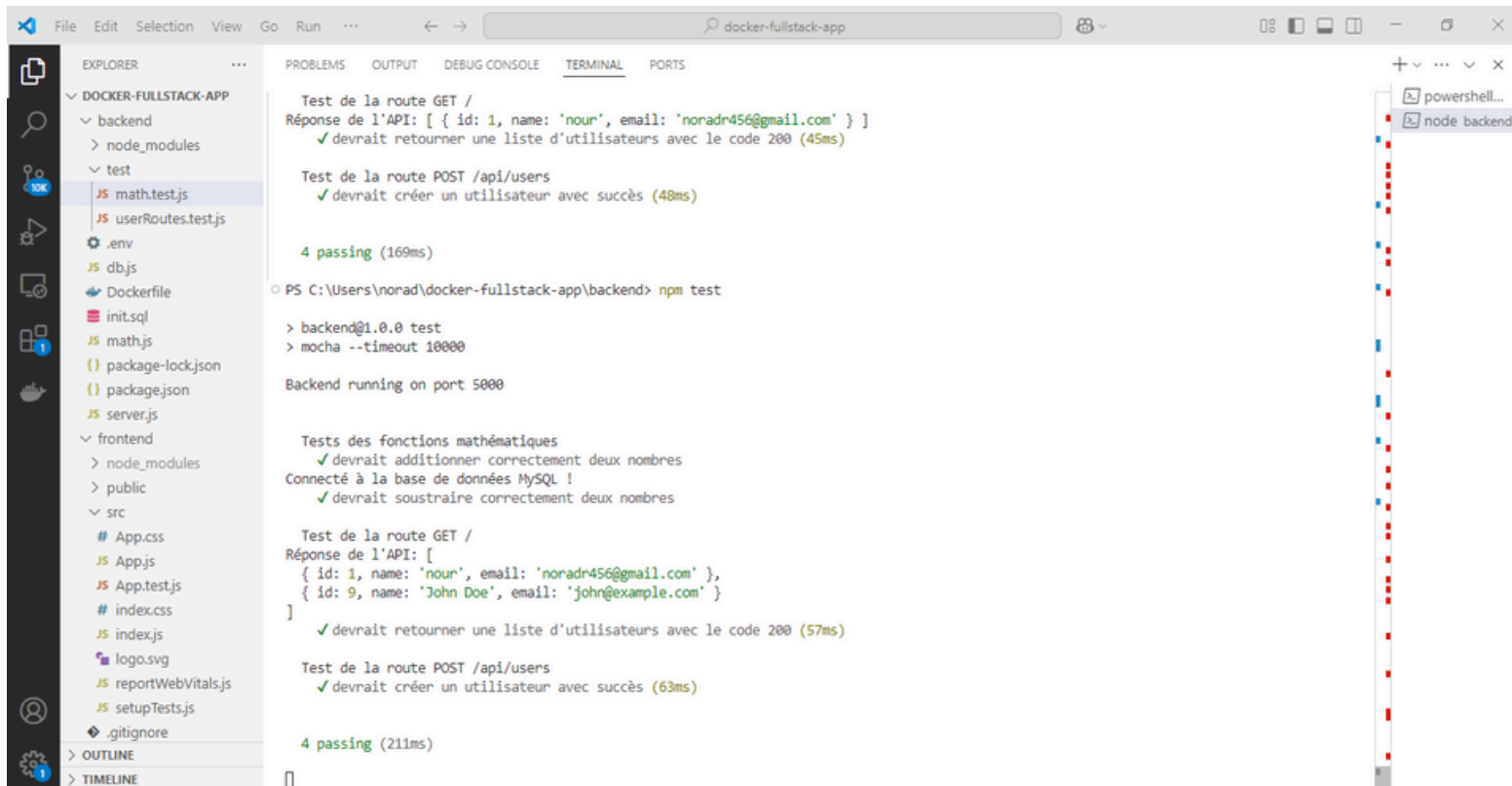
## DIFFICULTÉS RENCONTRÉES ET SOLUTIONS

- **Problèmes de tests avec Mocha/Chai** : Des problèmes de tests unitaires ont été rencontrés, notamment en raison:
  - Des erreurs lors du **require** ou **import** de certains modules.
  - Des conflits entre les versions de Mocha, Chai et les dernières versions de Node.js.
- **Solutions** :
  - Utilisation de Supertest pour effectuer les tests d'intégration sur les API.



# CAPTURES D'ÉCRAN

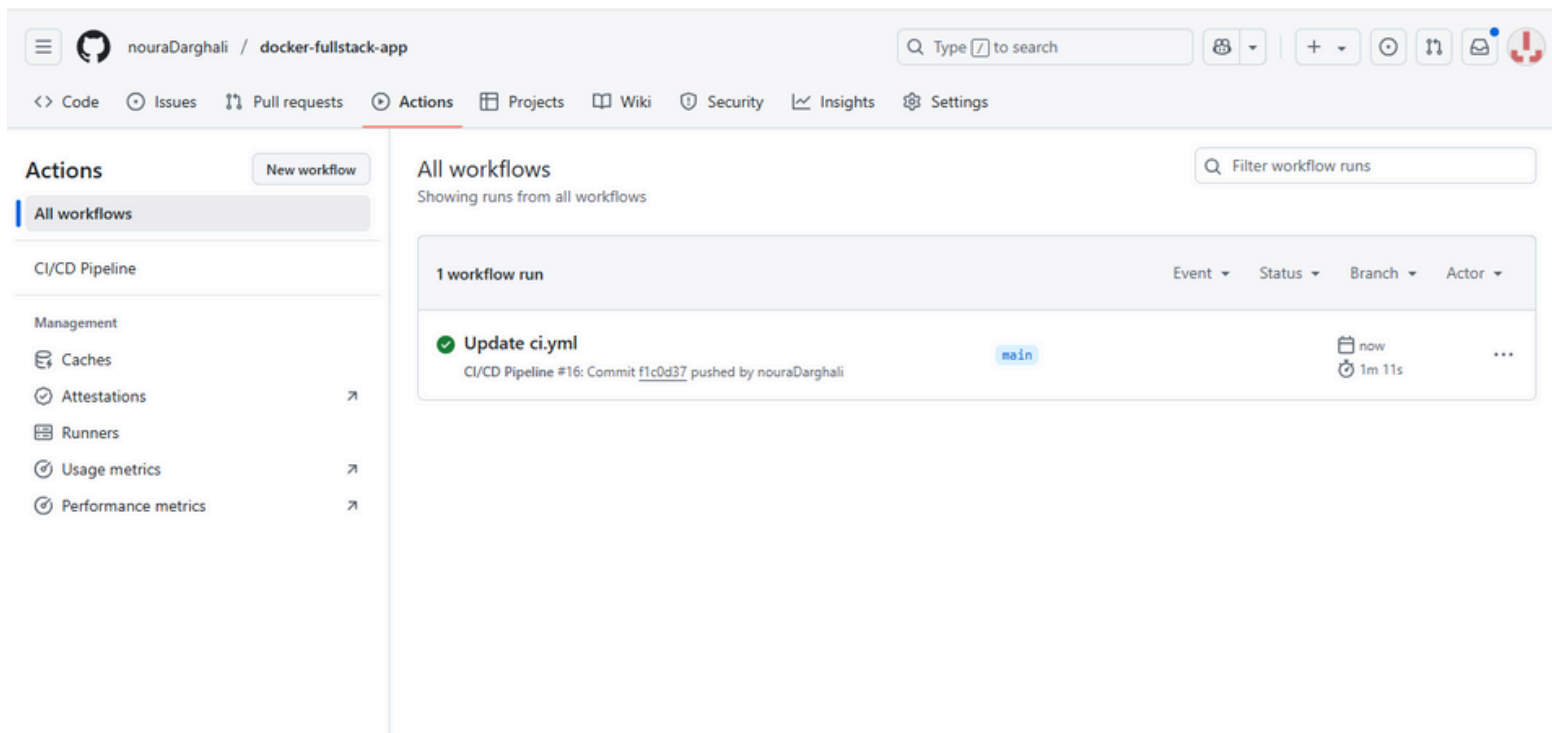
## LES TESTS



The screenshot shows the Visual Studio Code interface with the 'docker-fullstack-app' project open. The Explorer panel on the left shows the file structure, including 'backend' and 'test' folders. The Terminal panel on the right displays the output of running 'npm test'. The output shows that all tests passed, including tests for the GET / route, the POST /api/users route, and mathematical functions. The tests were run on a backend running on port 5000.

```
Test de la route GET /  
Réponse de l'API: [ { id: 1, name: 'nour', email: 'noradr456@gmail.com' } ]  
✓ devrait retourner une liste d'utilisateurs avec le code 200 (45ms)  
  
Test de la route POST /api/users  
✓ devrait créer un utilisateur avec succès (48ms)  
  
4 passing (169ms)  
  
PS C:\Users\norad\docker-fullstack-app\backend> npm test  
  
> backend@1.0.0 test  
> mocha --timeout 10000  
  
Backend running on port 5000  
  
Tests des fonctions mathématiques  
✓ devrait additionner correctement deux nombres  
Connecté à la base de données MySQL !  
✓ devrait soustraire correctement deux nombres  
  
Test de la route GET /  
Réponse de l'API: [  
  { id: 1, name: 'nour', email: 'noradr456@gmail.com' },  
  { id: 9, name: 'John Doe', email: 'john@example.com' }  
> ]  
✓ devrait retourner une liste d'utilisateurs avec le code 200 (57ms)  
  
Test de la route POST /api/users  
✓ devrait créer un utilisateur avec succès (63ms)  
  
4 passing (211ms)
```

## ACTIONS GITHUB



The screenshot shows the GitHub Actions page for the 'docker-fullstack-app' repository. The 'Actions' tab is selected, showing a list of workflows. The 'CI/CD Pipeline' workflow is highlighted, and its details are shown on the right. The workflow is triggered by a push to the 'main' branch and is currently in a 'pending' state. The workflow run is titled 'Update ci.yml' and was pushed by 'nouraDarghali'.

Actions

New workflow

All workflows

CI/CD Pipeline

Management

Caches

Attestations

Runners

Usage metrics

Performance metrics

All workflows

Showing runs from all workflows

Filter workflow runs

1 workflow run

Event Status Branch Actor

Update ci.yml

CI/CD Pipeline #16: Commit f1c0d37 pushed by noutraDarghali

main

now

1m 11s

github.com/nouraDarghali/docker-fullstack-app/actions/runs/14365596672/job/40277582926

nouraDarghali / docker-fullstack-app

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

CI/CD Pipeline

Update ci.yml #16

Summary

Jobs

build

Run details

Usage

Workflow file

build

succeeded tomorrow in 1m 7s

Search logs

- Set up job 1s
- Initialize containers 25s
- Checkout code 2s
- Setup Node.js 2s
- Wait for MySQL to be ready 20s
- Install dependencies 3s
- Build Docker image 8s
- Push Docker image to Docker Hub 4s
- Post Setup Node.js 0s
- Post Checkout code 0s
- Stop containers 1s
- Complete job 0s

# CONTENEURS DOCKER

docker.desktop PERSONAL

Search

Ctrl+K

Sign in

Sign in to use additional features enabled by your organization.

Containers

Images

Volumes

Builds

Docker Hub

Docker Scout

Extensions

Containers [Give feedback](#)

View all your running containers and applications. [Learn more](#)

Container CPU usage 1.01% / 400% (4 CPUs available)

Container memory usage 452.15MB / 3.69GB

Show charts

Search

Only show running containers

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	welcome-to-dock	4d17402a7eca	<a href="#">docker/welcome</a>	8088:80	0%	1 day ago	
<input type="checkbox"/>	docker-fullstack: -	-	-	-	1%	7 seconds ago	
<input type="checkbox"/>	frontend-1	d7d8f9151965	<a href="#">docker-fullstack: 3000:80</a>		0%	9 seconds ago	
<input type="checkbox"/>	backend-1	63ca30823b53	<a href="#">docker-fullstack: 5000:5000</a>		0%	7 seconds ago	
<input type="checkbox"/>	phpmyadmin-	dd22d71ca901	<a href="#">phpmyadmin/php: 8080:80</a>		0%	4 hours ago	
<input type="checkbox"/>	mysql_db-1	394c34995d16	<a href="#">mysql:8.0</a>	<a href="#">3306:3306</a>	1%	4 hours ago	

Engine running

RAM 1.28 GB CPU 0.00% Disk: 22.96 GB used (limit 1006.85 GB)

Terminal

New version available

Activier Windows

Accédez aux paramètres pour activer Windows.

Showing 6 items

# DOCKER HUB

Explore

My Hub

Search Docker Hub

CtrlK

N

nouradarghali  
Docker Personal

Repositories

Settings

Default privacy

Notifications

Billing

Usage

Pulls

Storage

Repositories / fullstack-app / General

Using 0 of 1 private repositories. [Get more](#)

nouradarghali/fullstack-app

Last pushed in 1 day • Repository size: 51.5 MB

Add a description

Add a category

General

Tags

Image Management

Collaborators

Webhooks

Settings

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest		Image	1 day	1 day

[See all](#)

DOCKER SCOUT INACTIVE

Activate

buildcloud

Build with Docker Build Cloud

Accelerate image build times with access to cloud-based builders and shared cache.

Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

Public view

```
docker push nouradarghali/fullstack-app:tagname
```

# CONCLUSION ET AXES D'AMÉLIORATION

Ce projet fullstack a permis de mettre en place une application web complète, allant de la gestion des utilisateurs à travers une interface frontend moderne, jusqu'à l'implémentation d'un backend robuste avec Node.js, Express et une base de données MySQL. L'intégration de Docker et l'automatisation via GitHub Actions ont enrichi le projet d'une dimension DevOps, garantissant portabilité, reproductibilité et automatisation du déploiement.

## Axes d'amélioration :

- Optimisation des requêtes SQL pour améliorer la performance sur de grandes bases de données.
- Ajout de fonctionnalités avancées comme la gestion des rôles et des permissions pour les utilisateurs.
- Déploiement sur un serveur cloud pour un accès à grande échelle et une meilleure sécurité.

## Annexe : Quelques prompts utilisés avec ChatGPT

- Pour résoudre un bug de test :

"J'ai une erreur avec Mocha/Chai lors d'un test unitaire, probablement liée à la version de Node.js. Que puis-je faire ?"

- Pour comprendre un concept :

"Quelle est la différence entre un test unitaire et un test d'intégration ?"

- Pour créer un pipeline GitHub Actions :

"Aide-moi à écrire un fichier ci.yml pour lancer les tests, builder une image Docker, et la pousser sur Docker Hub."