Justine GUIRAUDEN & Volcy DESMAZURES

Mini To-Do App - Projet DevOps

Introduction

Ce projet est une mini application To-Do développée dans le cadre du projet final DevOps. L'objectif principal est de démontrer la mise en place d'une chaîne CI/CD complète, depuis le développement jusqu'au déploiement automatique sur Kubernetes.

Le projet met l'accent sur :

- l'automatisation,
- la reproductibilité,
- les bonnes pratiques DevOps.

Architecture globale

L'architecture du projet repose sur une séparation claire des responsabilités :

- Frontend : interface utilisateur simple (HTML / JavaScript)
- Backend : API REST FastAPI
- Base de données :
    - SQLite en local
    - PostgreSQL lors du déploiement Kubernetes
- CI/CD : GitHub Actions
- Conteneurisation : Docker
- Orchestration : Kubernetes (kind)

Technologies utilisées

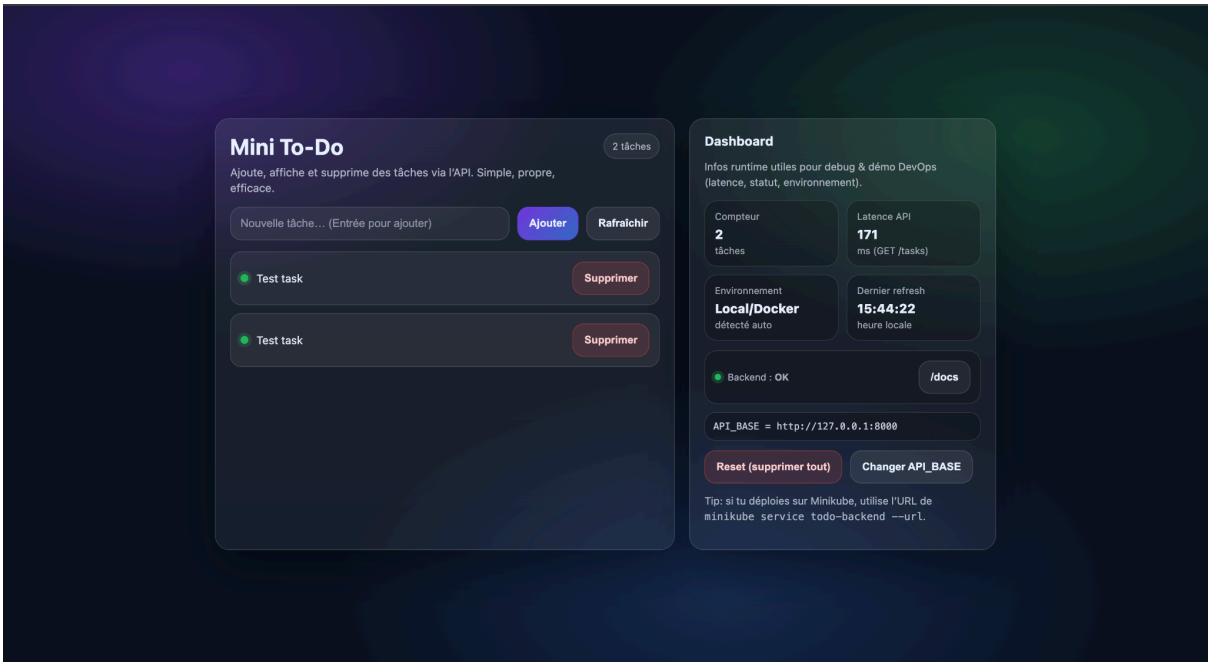
Composant	Technologie
Backend	FastAPI (Python)
Frontend	HTML / JavaScript
Bas de données	SQLite / PostgreSQL
Conteneurs	Docker
CI/CD	GitHub Actions

Justine GUIRAUDEN & Volcy DESMAZURES

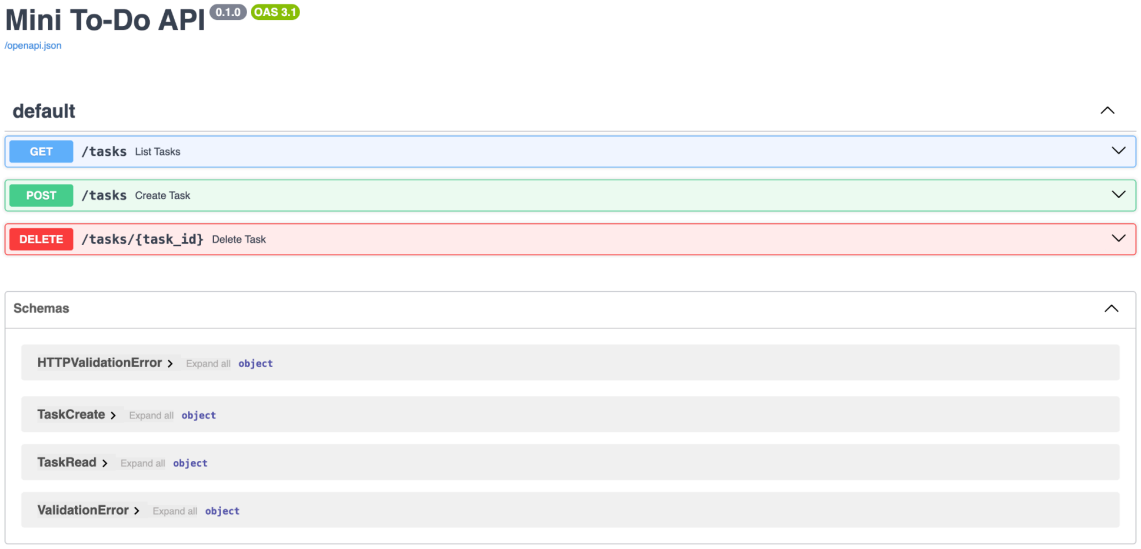
Orchestration	Kubernetes
Registry	Docker

Fonctionnalités de l’application

- Ajouter une tâche
- Supprimer une tâche
- Afficher la liste des tâches
- API REST documentée via Swagger
- Interface web simple
- Persistance des données
- Déploiement automatisé



Titre : Interface web de l’application (frontend)



Titre : Swagger UI du backend (/docs)

Dockerisation

L'application est entièrement conteneurisée.

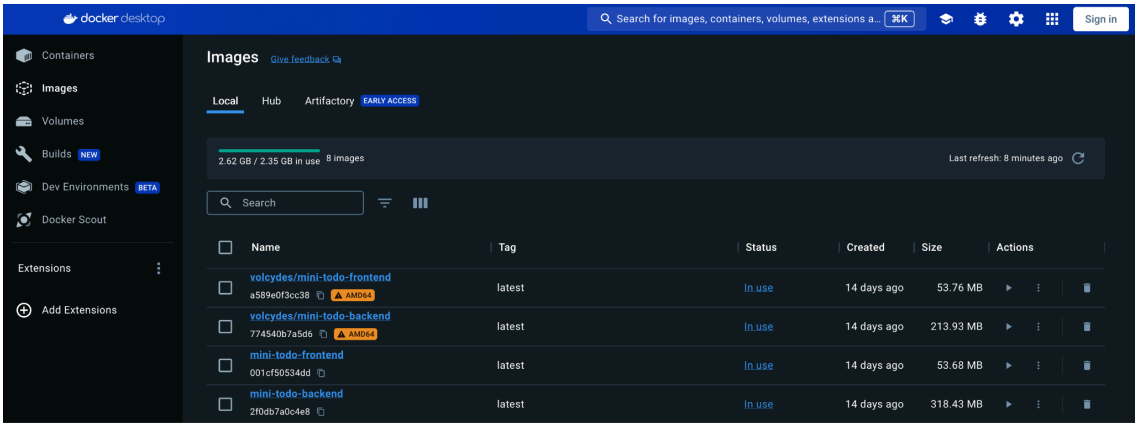
Backend

- Image Docker construite automatiquement

Frontend

- Image Docker construite automatiquement

Les images sont publiées automatiquement sur Docker Hub via la pipeline CI.



Titre : Page Docker Hub montrant les images backend et frontend

Justine GUIRAUDEN & Volcy DESMAZURES

Lancement de l’application

Option 1 — Lancement avec Docker (recommandé)

```
docker run -p 8000:8000 volcydes/mini-todo-backend:latest
docker run -p 8080:80 volcydes/mini-todo-frontend:latest
```

- Backend : http://localhost:8000/docs
- Frontend : http://localhost:8080

Cette méthode permet de lancer l’application sans configuration supplémentaire.

Option 2 — Lancement en local (développement)

Backend

```
cd backend
pip install -r requirements.txt
uvicorn app:app --reload
```

Frontend

```
cd frontend
open index.html
```

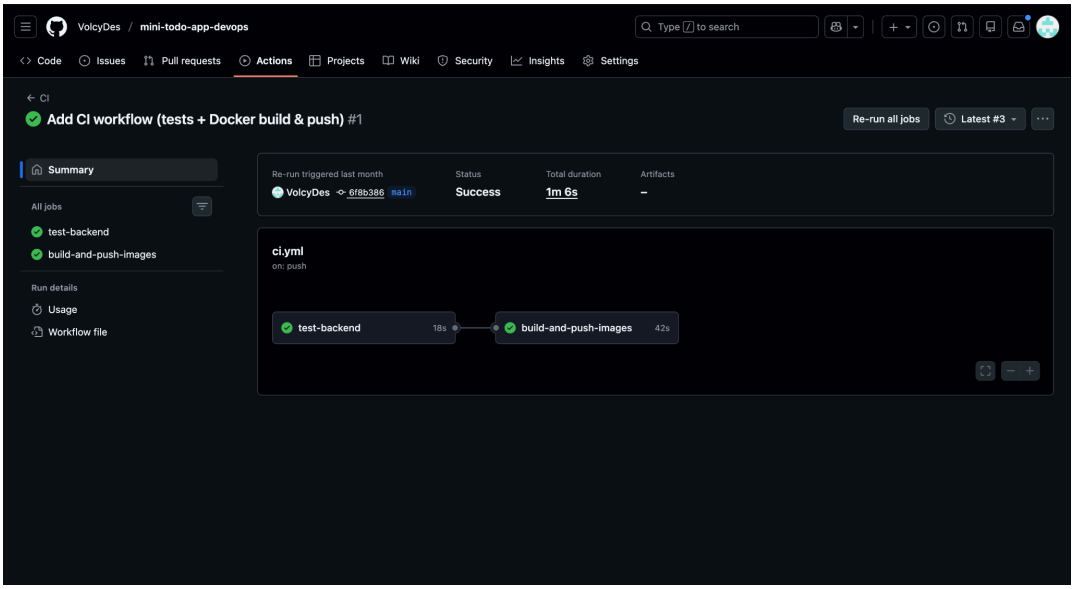
CI — Continuous Integration

La pipeline CI est déclenchée automatiquement à chaque push sur la branche `main`.

Étapes de la CI :

1. Récupération du code
2. Installation des dépendances Python
3. Exécution des tests automatisés (`pytest`)
4. Build des images Docker
5. Push des images sur Docker Hub

Justine GUIRAUDEN & Volcy DESMAZURES



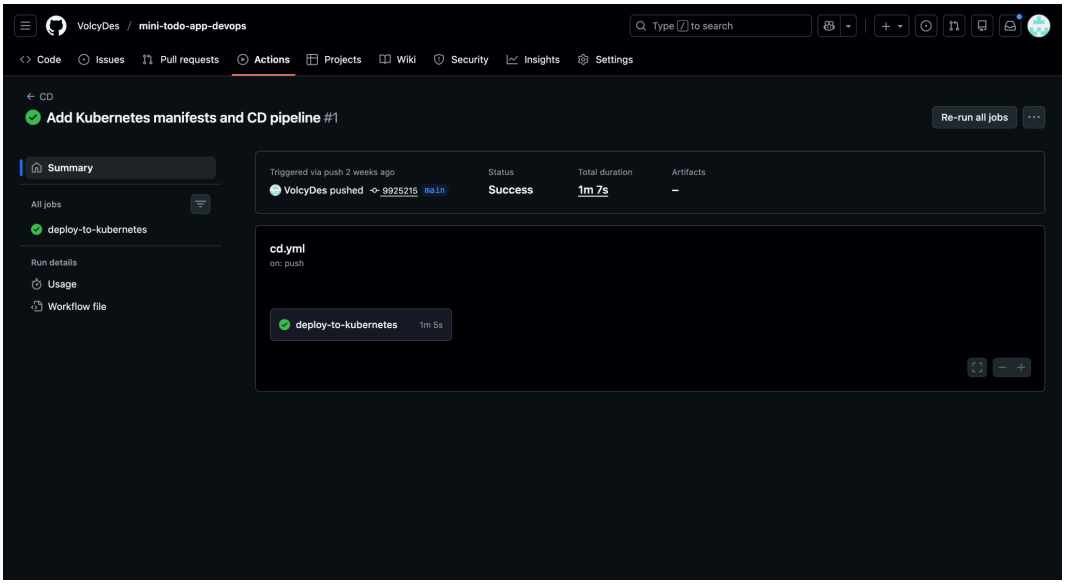
Titre : Pipeline CI GitHub Actions en succès (étapes vertes

CD — Continuous Deployment (Kubernetes)

La pipeline CD complète la chaîne CI/CD en assurant le déploiement automatique.

Fonctionnement de la CD :

- Déclenchée à chaque push sur `main`
- Création d'un cluster Kubernetes éphémère via kind
- Application automatique des manifests Kubernetes
- Vérification de l'état des pods et services



Titre : Pipeline CD GitHub Actions en succès

Justine GUIRAUDEN & Volcy DESMAZURES

Kubernetes

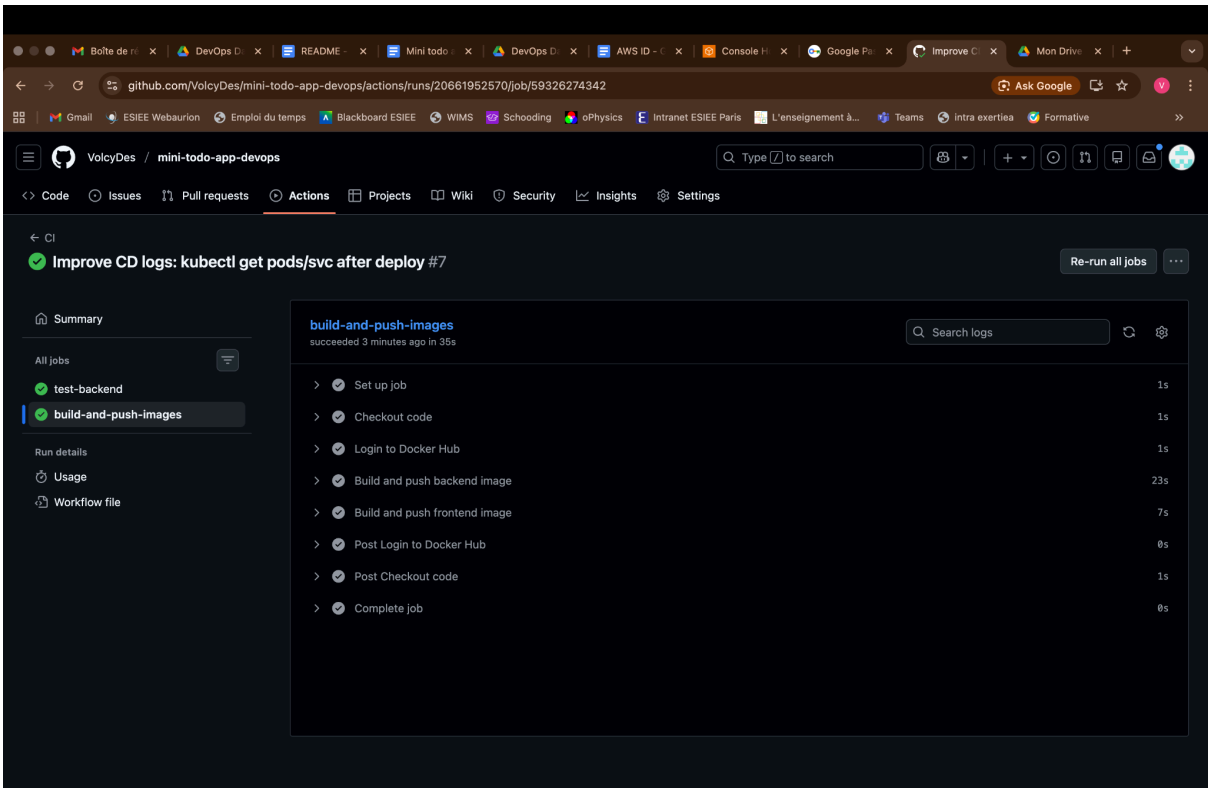
Les manifests Kubernetes sont regroupés dans le dossier :

```
k8s/
├── postgres.yml
└── backend.yml
```

Déploiements inclus :

- PostgreSQL :
  - Secret
  - PersistentVolumeClaim
  - Deployment
  - Service
- Backend FastAPI :
  - Deployment
  - Service NodePort

Le déploiement est effectué automatiquement par la pipeline CD.



Titre : Pipeline CI – Build & Push des images Docker (succès)

Conclusion

Ce projet démontre la mise en œuvre complète d'une chaîne DevOps moderne, incluant :

- le développement applicatif,
- la conteneurisation,
- l'intégration continue,
- le déploiement continu sur Kubernetes.

L'ensemble du processus est automatisé et reproductible, répondant aux exigences du projet.