

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)

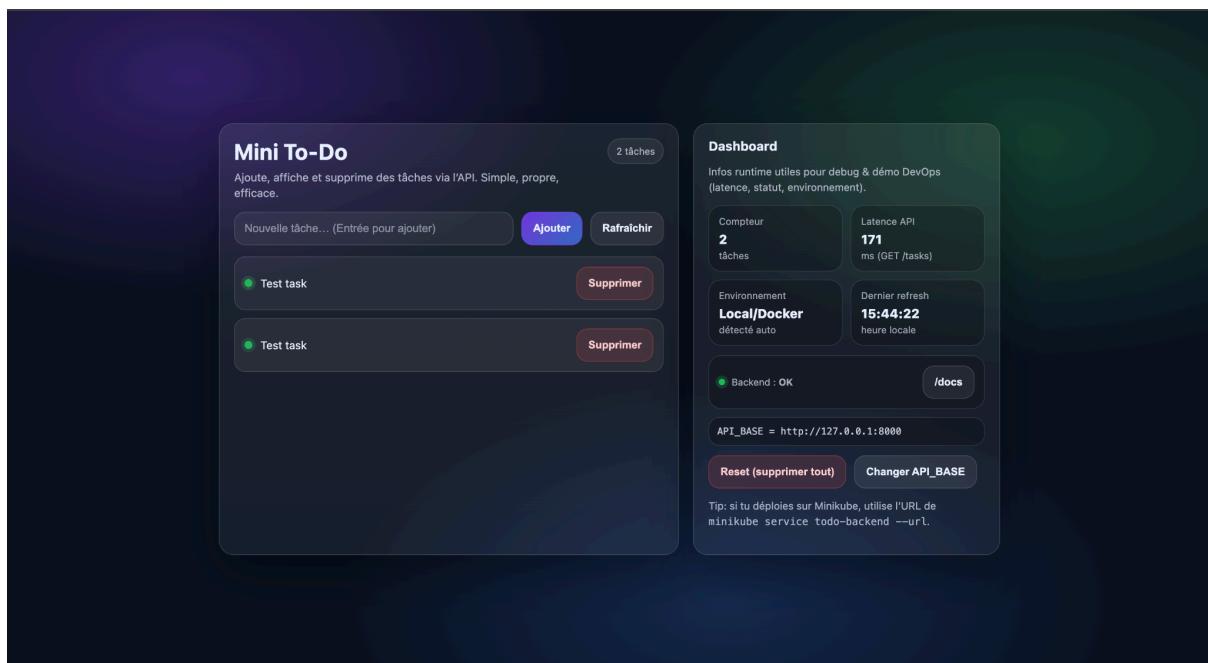
Technologies utilisées

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

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)

The screenshot shows the Swagger UI interface for a 'Mini To-Do API'. At the top, it displays the version '0.1.0' and 'OAS 3.1'. Below this, under the 'default' endpoint, there are three operations: a blue 'GET /tasks' button labeled 'List Tasks', a green 'POST /tasks' button labeled 'Create Task', and a red 'DELETE /tasks/{task_id}' button labeled 'Delete Task'. Each operation has a small downward arrow icon to its right. Below the endpoint, there is a section titled 'Schemas' which lists several schema definitions: 'HTTPValidationErrors > Expand all object', 'TaskCreate > Expand all object', 'TaskRead > Expand all object', and 'ValidationError > Expand all object'. Each schema entry has a small downward arrow icon to its right.

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.

The screenshot shows the Docker Desktop interface with the 'Images' tab selected. On the left sidebar, there are icons for 'Containers', 'Images' (which is selected and highlighted in blue), 'Volumes', 'Builds' (marked as 'NEW'), 'Dev Environments' (marked as 'BETA'), and 'Docker Scout'. The main area shows a table of images with the following data:

Name	Tag	Status	Created	Size	Actions
volcydes/mini-todo-frontend	latest	In use	14 days ago	53.76 MB	[...]
a589e0f3cc38	AMD64				
volcydes/mini-todo-backend	latest	In use	14 days ago	213.93 MB	[...]
774540b7a5d6	AMD64				
mini-todo-frontend	latest	In use	14 days ago	53.68 MB	[...]
001cf5054d40					
mini-todo-backend	latest	In use	14 days ago	318.43 MB	[...]
2f0db7a0c4e8					

Titre : Page Docker Hub montrant les images backend et frontend

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

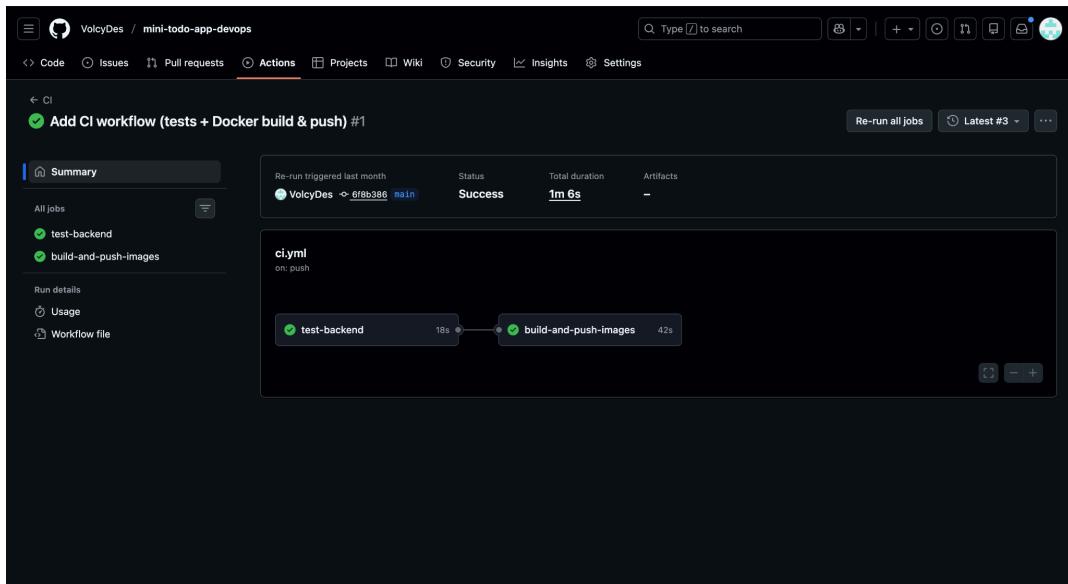
```
cd frontend
open index.html
```

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



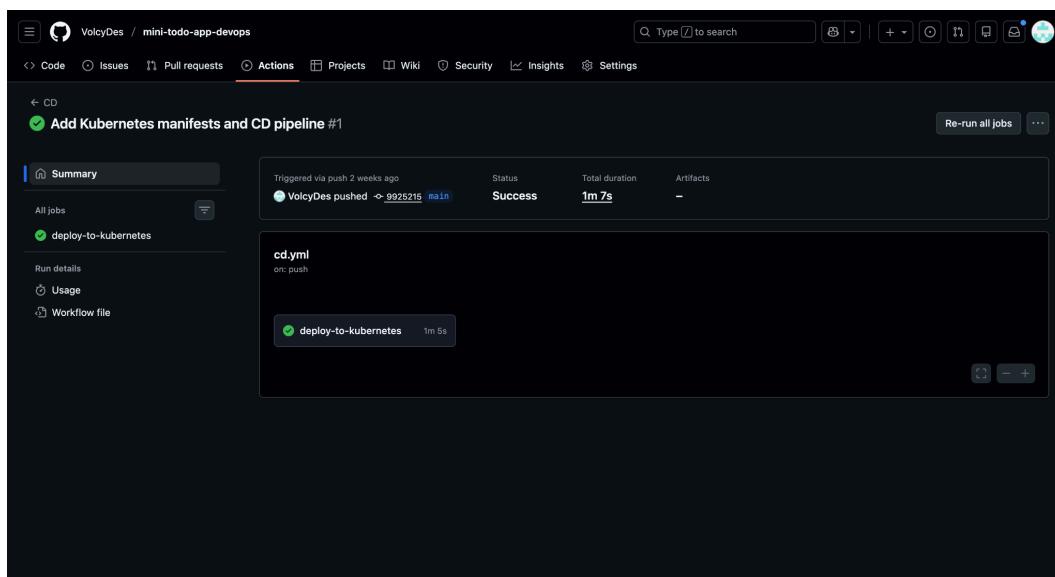
Titre : Pipeline CI GitHub Actions en succès (étapes vertes)

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

Kubernetes

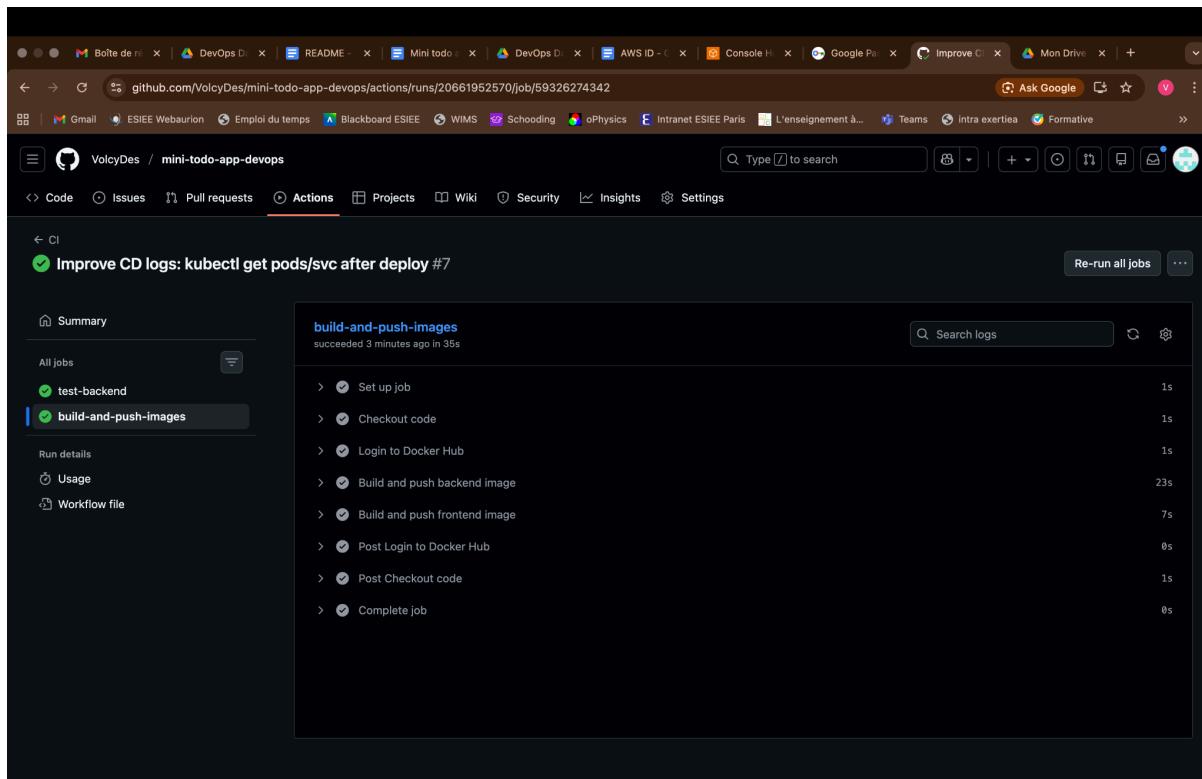
Les manifests Kubernetes sont regroupés dans le dossier :

```
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 CI.



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.