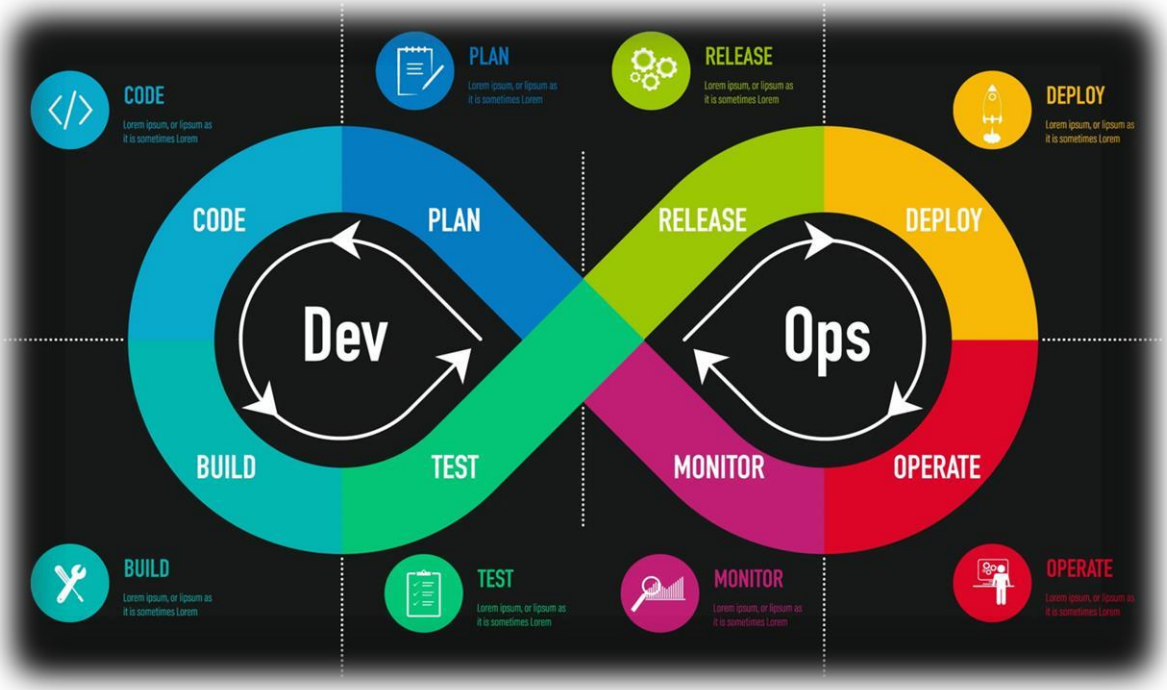
Rapport de Projet : IoT Monitor – MicroServices DevOps



Étudiante : Celia MERABET

URL du dépôt GitHub : https://github.com/celia-merabet/iotmonitoring-devops

Table des matières :

1. Présentation du Projet

Le projet **IoT Monitor** est une plateforme de monitoring de capteurs IoT.
L'objectif est de mettre en place une **infrastructure micro-services conteneurisée avec Docker**, sécurisée et accessible via un reverse proxy et un tunnel Cloudflare.

Fonctionnalités principales :

- Dashboard web pour visualiser les données de capteurs IoT
- API backend pour la réception des données simulées
- Base de données PostgreSQL avec persistance des données
- Interface Adminer pour la gestion de la base de données
- Accès HTTPS public via Cloudflare Tunnel (optionnel)

L'accent est mis sur l'infrastructure et la robustesse des services, pas sur des fonctionnalités avancées IoT.

2. Architecture Technique

2.1 Schéma PlantUML

```
@startuml !theme
sunlust
title Architecture DevOps du Projet IoT Monitor

node "Docker Host" {
    component "Cloudflared" as Tunnel #Orange
component "Caddy" as Proxy #Yellow

    package "Internal Network" {
        component "Frontend Nginx" as Frontend #LightBlue
component "API Node.js" as API #LightGreen
component "PostgreSQL DB" as DB #Pink         component
"Adminer" as Admin #LightGray     }
}
cloud "Internet" {
actor User
}

User --> Tunnel : HTTPS
Tunnel --> Proxy : HTTP
Proxy --> Frontend : /
Proxy --> API : /api/*
Proxy --> Admin : /admin
Frontend --> API : Requêtes API
API --> DB : Lecture / Écriture
Admin --> DB : Administration @enduml
```

2.2 Description des services

Service	Image Docker	Rôle
Proxy	caddy:latest	Reverse Proxy & Routage
Frontend	nginx:alpine	Dashboard Web IoT
API	node:18-alpine	API de collecte des données
DB	postgres:15	Base de données persistante
Admin	adminer	Administration BDD
Cloudflared	cloudflare/cloudflared	Tunnel HTTPS public N/A

Tous les conteneurs sont configurés avec `restart: unless-stopped` et healthchecks pour assurer la robustesse.

3. Guide d'installation

3.1 Prérequis

- Docker Desktop (Mac, Windows ou Linux)
- Docker Compose 3.9+
- Compte Cloudflare pour le tunnel (optionnel pour accès public)

3.2 Installation

1. Cloner le dépôt :

```
git clone https://github.com/celia-merabet/iot-monitoring-devops.git cd
iot-monitoring-devops
```

2. Lancer la stack (local) :

```
docker compose up -d --build
```

3. Vérifier les conteneurs :

```
docker ps
```

4. Accéder aux services :

Service	URL
Frontend	http://localhost
API	http://localhost:3000/api/statusAdminer http://localhost:8080

5. Si le tunnel Cloudflare est configuré, obtenir l'URL publique : `docker compose logs -f cloudflared`

4. Méthodologie & Organisation

- Projet réalisé en autonomie.
- Approche progressive : création du Dockerfile pour l'API → conteneurisation frontend → configuration BDD → reverse proxy Caddy → Cloudflare Tunnel.
- Tests réguliers via `curl` et navigateur pour vérifier la connectivité interne et externe.

5. Utilisation de l'IA

Outils utilisés : capilote

Usage :

- Conseils ponctuels de syntaxe et configuration Docker
- Débogage des conteneurs (API Node.js et Adminer)
- Suggestions pour l'architecture PlantUML

Apprentissage : L'IA a été utilisée comme assistant ; toutes les configurations ont été testées et adaptées manuellement par moi.

6. Difficultés rencontrées & Solutions

Problème	Solution
API Node.js non accessible depuis Mac	Ajout d'un mapping de port `3000:3000`
Adminer inaccessible	Ajout d'un mapping de port `8080:8080` et choix du système PostgreSQL
Cloudflared restart en boucle	Token non configuré → tunnel commenté temporairement
Perte de données après redémarrage	Création d'un volume Docker pour la BDD

7. Résultats attendus

- Frontend accessible sur http://localhost
- API accessible sur http://localhost:3000/api/status
- Base PostgreSQL persistante via volume

- Adminer accessible sur http://localhost:8080
- Architecture robuste avec restart policies et isolation réseau

8. Conclusion

Ce projet montre la mise en place d'une **stack micro-services complète** avec Docker, intégrant :

- Frontend et backend séparés Base de données persistante Interface d'administration
- Reverse proxy Caddy

9. Annexes

- **Schéma PlantUML** : `architecture.puml`
- **Frontend HTML minimal** : `frontend/index.html`
- **API Node.js** : `api/index.js`
- **Docker Compose** : `docker-compose.yml`
- **Caddy configuration** : `Caddyfile`

10. Ressources & Liens Utiles

Documentation officielle

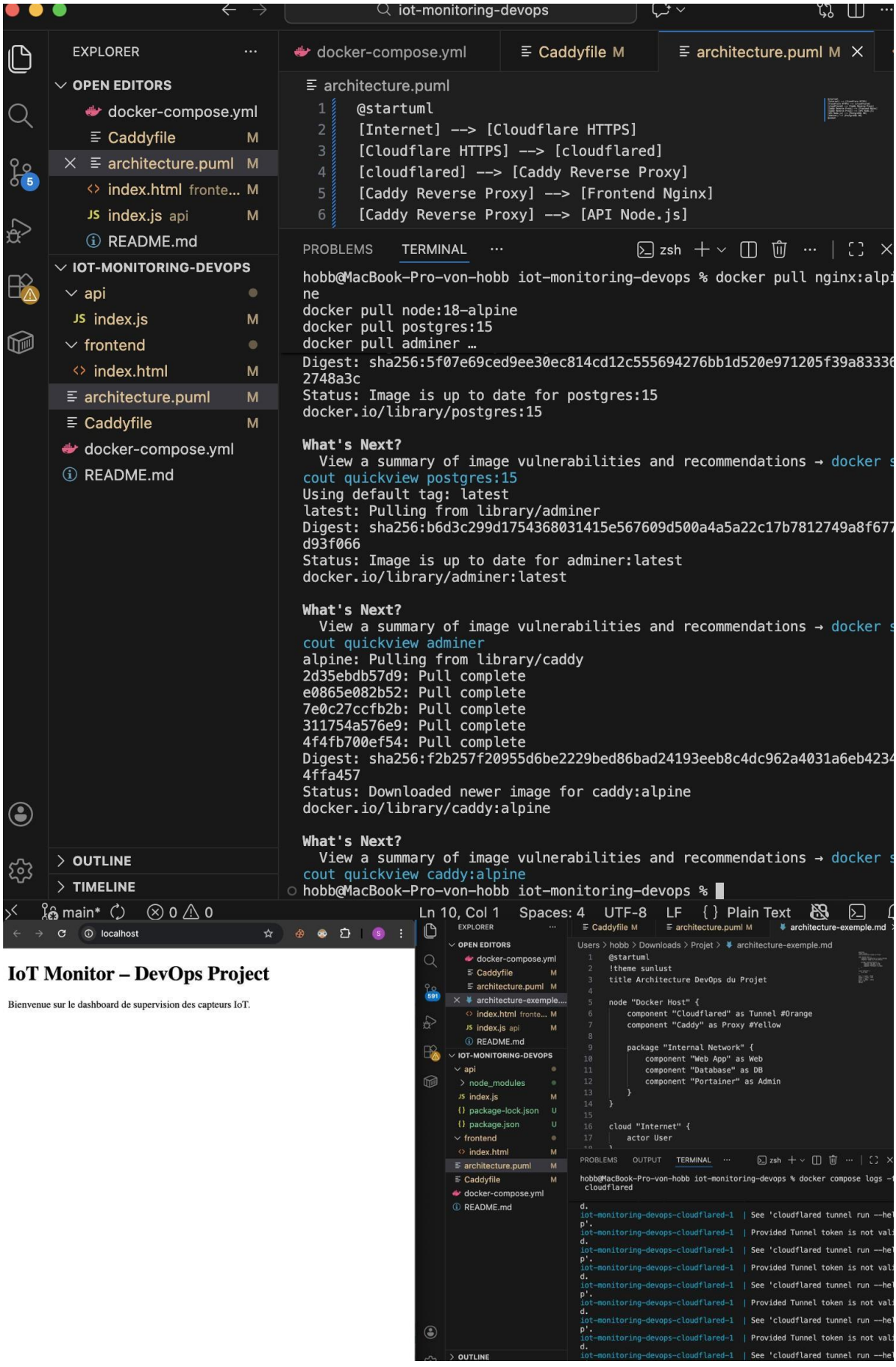
- **Docker** : https://docs.docker.com/
- **Docker Compose** : https://docs.docker.com/compose/
- **Node.js** : https://nodejs.org/en/docs/
- **PostgreSQL** : https://www.postgresql.org/docs/
- **Adminer** : https://www.adminer.org/
- **Caddy (reverse proxy)** : https://caddyserver.com/docs/
- **Cloudflare Tunnel (cloudflared)** : https://developers.cloudflare.com/cloudflareone/connections/connect-apps/

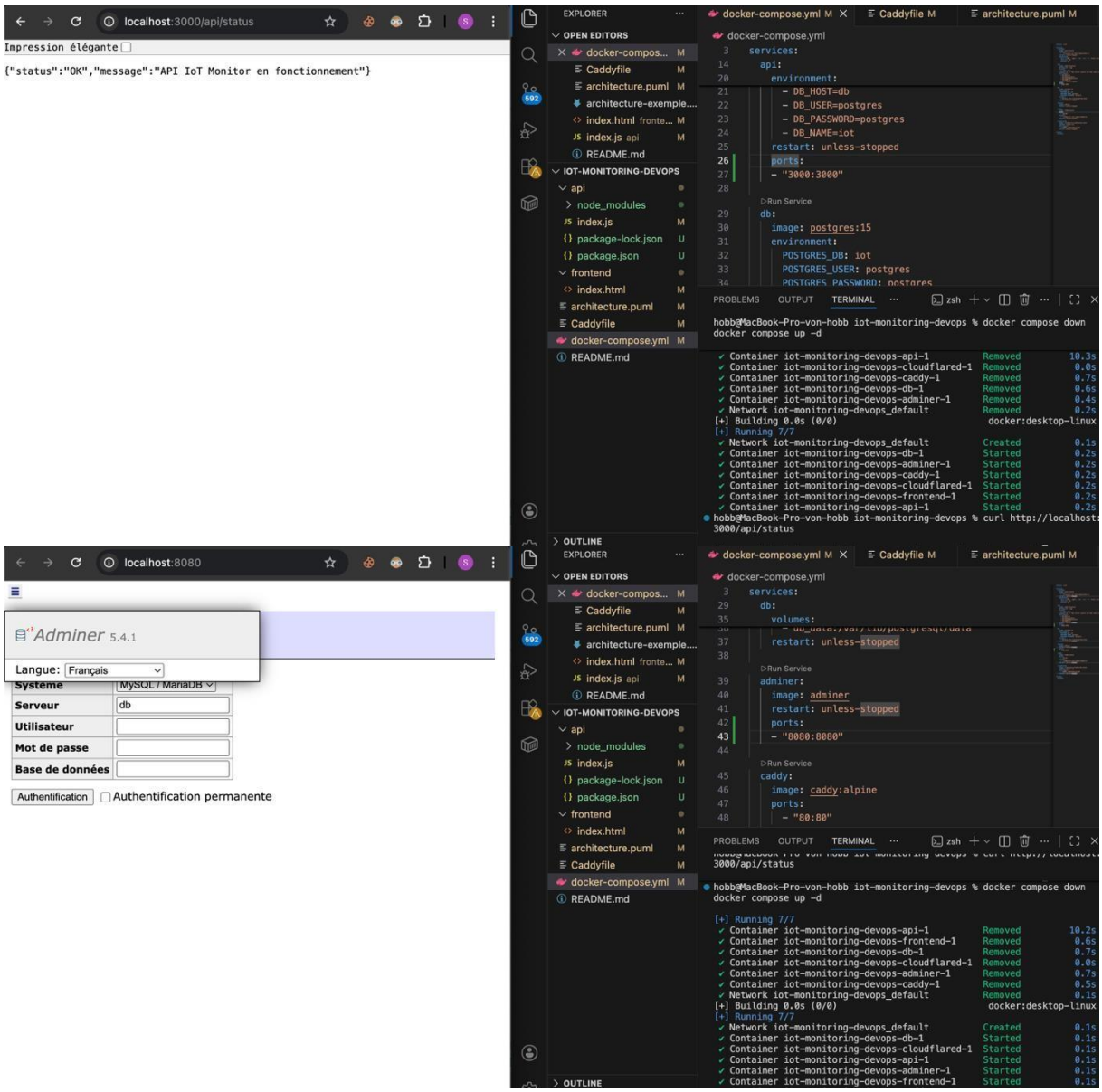
Tutoriels et guides utiles

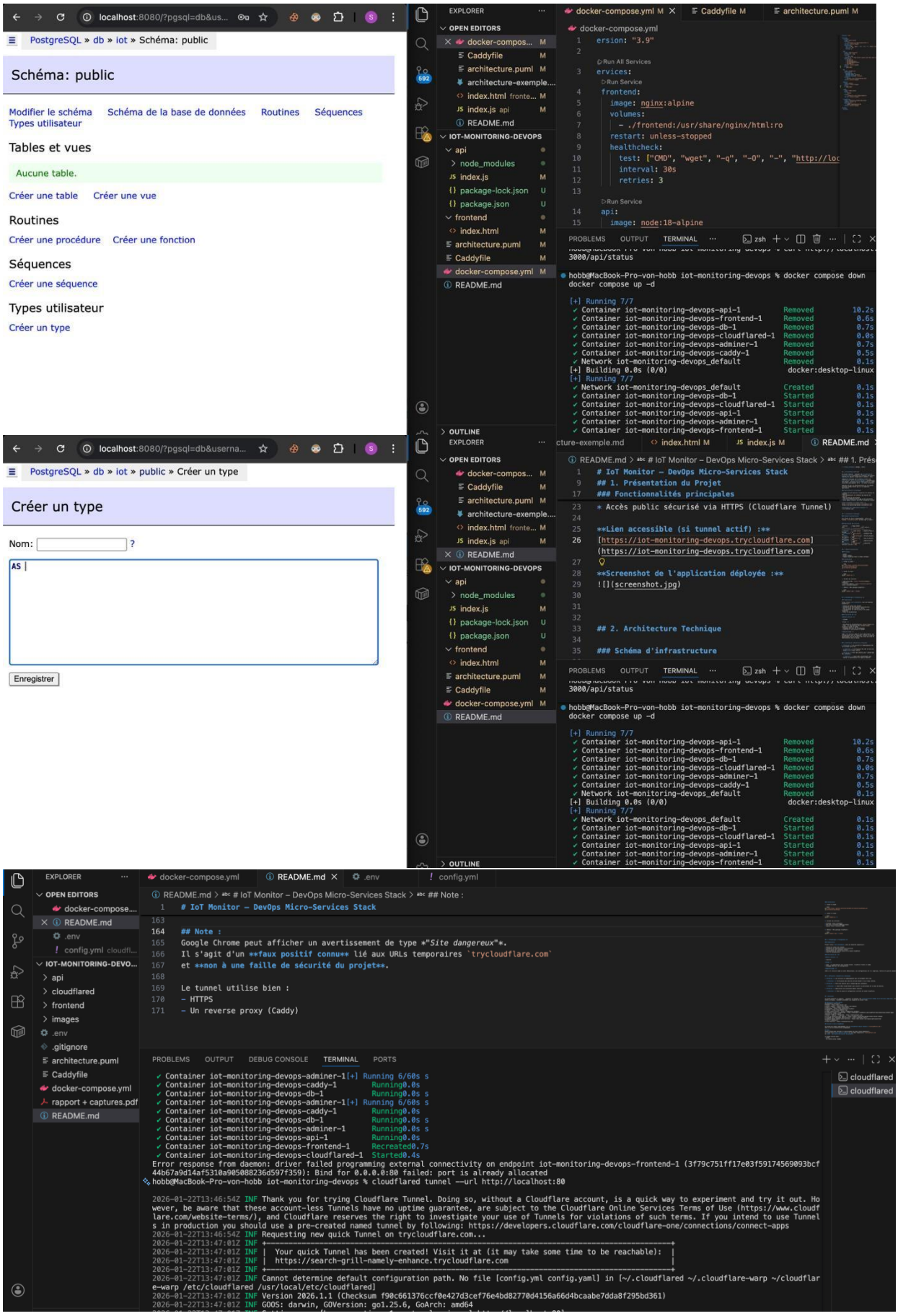
- **Docker + Node.js + PostgreSQL** : https://node-postgres.com/
- **Créer un Dockerfile pour Node.js** : https://nodejs.org/en/docs/guides/nodejs-dockerwebapp/
- **PlantUML pour diagrammes d'architecture** : https://plantuml.com/fr/ **Exemples Docker Compose multi-services** : https://docs.docker.com/compose/gettingstarted/

Forums et communautés

- **Stack Overflow** : https://stackoverflow.com/


iot-monitoring-devops
EXPLORER
OPEN EDITORS
docker-compose.yml
Caddyfile M
architecture.puml M
index.html fronte... M
index.js api M
README.md
IOT-MONITORING-DEVOPS
api
index.js M
frontend
index.html M
architecture.puml M
Caddyfile M
docker-compose.yml
README.md
OUTLINE
TIMELINE
docker-compose.yml
Caddyfile M
architecture.puml M
architecture.puml
1 @startuml
2 [Internet] --> [Cloudflare HTTPS]
3 [Cloudflare HTTPS] --> [cloudflared]
4 [cloudflared] --> [Caddy Reverse Proxy]
5 [Caddy Reverse Proxy] --> [Frontend Nginx]
6 [Caddy Reverse Proxy] --> [API Node.js]
PROBLEMS
TERMINAL
zsh
hobb@MacBook-Pro-von-hobb iot-monitoring-devops % docker pull nginx:alpine
docker pull node:18-alpine
docker pull postgres:15
docker pull adminer …
Digest: sha256:5f07e69ced9ee30ec814cd12c555694276bb1d520e971205f39a83332748a3c
Status: Image is up to date for postgres:15
docker.io/library/postgres:15
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview postgres:15
Using default tag: latest
latest: Pulling from library/adminer
Digest: sha256:b6d3c299d1754368031415e567609d500a4a5a22c17b7812749a8f677d93f066
Status: Image is up to date for adminer:latest
docker.io/library/adminer:latest
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview adminer
alpine: Pulling from library/caddy
2d35ebdb57d9: Pull complete
e0865e082b52: Pull complete
7e0c27ccfb2b: Pull complete
311754a576e9: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:f2b257f20955d6be2229bed86bad24193eeb8c4dc962a4031a6eb42344ffa457
Status: Downloaded newer image for caddy:alpine
docker.io/library/caddy:alpine
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview caddy:alpine
hobb@MacBook-Pro-von-hobb iot-monitoring-devops %
main*
0 0
Ln 10, Col 1
Spaces: 4
UTF-8
LF
Plain Text
localhost
IoT Monitor – DevOps Project
Bienvenue sur le dashboard de supervision des capteurs IoT.
architecture-exemple.md
@startuml
!theme sunlust
title Architecture DevOps du Projet
node "Docker Host" {
component "Cloudflared" as Tunnel #Orange
component "Caddy" as Proxy #Yellow
package "Internal Network" {
component "Web App" as Web
component "Database" as DB
component "Portainer" as Admin
}
}
cloud "Internet" {
actor User
PROBLEMS
OUTPUT
TERMINAL
hobb@MacBook-Pro-von-hobb iot-monitoring-devops % docker compose logs -f cloudflared
iot-monitoring-devops-cloudflared-1 | See 'cloudflared tunnel run --help' for more command
iot-monitoring-devops-cloudflared-1 | Provided Tunnel token is not valid.


localhost:3000/api/status
Impression élégante
{"status":"OK","message":"API IoT Monitor en fonctionnement"}
localhost:8080
Adminer 5.4.1
Langue: Français
Système
MySQL / MariaDB
Serveur
db
Utilisateur
Mot de passe
Base de données
Authentification
Authentification permanente
EXPLORER
OPEN EDITORS
IOT-MONITORING-DEVOPS
docker-compose.yml
Caddyfile
architecture.puml
README.md
PROBLEMS
OUTPUT
TERMINAL
hobb@MacBook-Pro-von-hobb iot-monitoring-devops % docker compose down
docker compose up -d
hobb@MacBook-Pro-von-hobb iot-monitoring-devops % curl http://localhost:3000/api/status


localhost:8080/?pgsql=db&us...
PostgreSQL » db » iot » Schéma: public
Schéma: public
Modifier le schéma
Schéma de la base de données
Routines
Séquences
Types utilisateur
Tables et vues
Aucune table.
Créer une table
Créer une vue
Routines
Créer une procédure
Créer une fonction
Séquences
Créer une séquence
Types utilisateur
Créer un type
localhost:8080/?pgsql=db&userna...
PostgreSQL » db » iot » public » Créer un type
Créer un type
Nom:
?
AS
Enregistrer
EXPLORER
OPEN EDITORS
docker-compos... M
Caddyfile M
architecture.puml M
architecture-exemple...
index.html fronte... M
index.js api M
README.md
IOT-MONITORING-DEVOPS
api
node_modules
index.js M
package-lock.json U
package.json U
frontend
index.html M
architecture.puml M
Caddyfile M
docker-compose.yml M
README.md
OUTLINE
docker-compose.yml M
Caddyfile M
architecture.puml M
docker-compose.yml
ersion: "3.9"
Run All Services
ervices:
Run Service
frontend:
image: nginx:alpine
volumes:
- ./frontend:/usr/share/nginx/html:ro
restart: unless-stopped
healthcheck:
test: ["CMD", "wget", "-q", "-O", "-", "http://loc
interval: 30s
retries: 3
Run Service
api:
image: node:18-alpine
PROBLEMS
OUTPUT
TERMINAL
zsh
3000/api/status
hobb@MacBook-Pro-von-hobb iot-monitoring-devops % docker compose down
docker compose up -d
[+] Running 7/7
✓ Container iot-monitoring-devops-api-1 Removed 10.2s
✓ Container iot-monitoring-devops-frontend-1 Removed 0.6s
✓ Container iot-monitoring-devops-db-1 Removed 0.7s
✓ Container iot-monitoring-devops-cloudflared-1 Removed 0.0s
✓ Container iot-monitoring-devops-adminer-1 Removed 0.7s
✓ Container iot-monitoring-devops-caddy-1 Removed 0.5s
✓ Network iot-monitoring-devops_default Removed 0.1s
[+] Building 0.0s (0/0) docker:desktop-linux
[+] Running 7/7
✓ Network iot-monitoring-devops_default Created 0.1s
✓ Container iot-monitoring-devops-db-1 Started 0.1s
✓ Container iot-monitoring-devops-cloudflared-1 Started 0.1s
✓ Container iot-monitoring-devops-api-1 Started 0.1s
✓ Container iot-monitoring-devops-adminer-1 Started 0.1s
✓ Container iot-monitoring-devops-frontend-1 Started 0.1s
cture-exemple.md
index.html M
index.js M
README.md
README.md > # IoT Monitor – DevOps Micro-Services Stack > ## 1. Prés
IoT Monitor – DevOps Micro-Services Stack
1. Présentation du Projet
Fonctionnalités principales
* Accès public sécurisé via HTTPS (Cloudflare Tunnel)
Lien accessible (si tunnel actif) :
[https://iot-monitoring-devops.trycloudflare.com]
(https://iot-monitoring-devops.trycloudflare.com)
Screenshot de l'application déployée :

2. Architecture Technique
Schéma d'infrastructure
EXPLORER
OPEN EDITORS
docker-compose....
README.md
.env
config.yml cloudfl...
IOT-MONITORING-DEVO...
api
cloudflared
frontend
images
.env
.gitignore
architecture.puml
Caddyfile
docker-compose.yml
rapport + captures.pdf
README.md
docker-compose.yml
README.md
.env
config.yml
README.md > # IoT Monitor – DevOps Micro-Services Stack > ## Note :
IoT Monitor – DevOps Micro-Services Stack
Note :
Google Chrome peut afficher un avertissement de type *"Site dangereux"*.
Il s'agit d'un **faux positif connu** lié aux URLs temporaires `trycloudflare.com`
et **non à une faille de sécurité du projet**.
Le tunnel utilise bien :
- HTTPS
- Un reverse proxy (Caddy)
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
cloudflared
cloudflared
✓ Container iot-monitoring-devops-adminer-1 [+] Running 6/60s s
✓ Container iot-monitoring-devops-caddy-1 Running 0.0s
✓ Container iot-monitoring-devops-db-1 Running 0.0s s
✓ Container iot-monitoring-devops-adminer-1 [+] Running 6/60s s
✓ Container iot-monitoring-devops-caddy-1 Running 0.0s
✓ Container iot-monitoring-devops-db-1 Running 0.0s s
✓ Container iot-monitoring-devops-adminer-1 Running 0.0s s
✓ Container iot-monitoring-devops-api-1 Running 0.0s
✓ Container iot-monitoring-devops-frontend-1 Recreated 0.7s
✓ Container iot-monitoring-devops-cloudflared-1 Started 0.4s
Error response from daemon: driver failed programming external connectivity on endpoint iot-monitoring-devops-frontend-1 (3f79c751ff17e03f59174569093bcf44b67a9d14af5310a905088236d597f359): Bind for 0.0.0.0:80 failed: port is already allocated
hobb@MacBook-Pro-von-hobb iot-monitoring-devops % cloudflared tunnel --url http://localhost:80
2026-01-22T13:46:54Z INF Thank you for trying Cloudflare Tunnel. Doing so, without a Cloudflare account, is a quick way to experiment and try it out. However, be aware that these account-less Tunnels have no uptime guarantee, are subject to the Cloudflare Online Services Terms of Use (https://www.cloudflare.com/website-terms/), and Cloudflare reserves the right to investigate your use of Tunnels for violations of such terms. If you intend to use Tunnels in production you should use a pre-created named tunnel by following: https://developers.cloudflare.com/cloudflare-one/connections/connect-apps
2026-01-22T13:46:54Z INF Requesting new quick Tunnel on trycloudflare.com...
2026-01-22T13:47:01Z INF
2026-01-22T13:47:01Z INF | Your quick Tunnel has been created! Visit it at (it may take some time to be reachable): |
2026-01-22T13:47:01Z INF | https://search-grill-namely-enhance.trycloudflare.com |
2026-01-22T13:47:01Z INF
2026-01-22T13:47:01Z INF Cannot determine default configuration path. No file [config.yml config.yaml] in [~/.cloudflared ~/.cloudflare-warp ~/cloudflare-warp /etc/cloudflared /usr/local/etc/cloudflared]
2026-01-22T13:47:01Z INF Version 2026.1.1 (Checksum f90c661376ccf0e427d3cef76e4bd82770d4156a66d4bcaabe7dda8f295bd361)
2026-01-22T13:47:01Z INF GOOS: darwin, GOVersion: go1.25.6, GoArch: amd64