Whanos – Images Docker

Auteur : (Alberic)

Objectif de la partie

Cette partie du projet Whanos consiste à **créer toutes les images Docker** nécessaires pour les langages supportés par l'infrastructure.
Ces images seront utilisées par **Jenkins (Patrick)** pour la phase d'intégration continue et par **Kubernetes (Maurel)** pour le déploiement automatique.

L'objectif est de permettre à n'importe quelle application "Whanos-compatible" d'être **containerisée automatiquement** à partir de sa technologie détectée.

Structure du répertoire

```
.
├── build_images.sh              # Script de construction de toutes les images
└── images/
    ├── befunge/
    │   ├── Dockerfile.base
    │   └── Dockerfile.standalone
    ├── c/
    │   ├── Dockerfile.base
    │   └── Dockerfile.standalone
    ├── java/
    │   ├── Dockerfile.base
    │   └── Dockerfile.standalone
    ├── javascript/
    │   ├── Dockerfile.base
    │   └── Dockerfile.standalone
    └── python/
        ├── Dockerfile.base
        └── Dockerfile.standalone
```

Chaque langage possède **deux images distinctes** :

- `Dockerfile.base` → image de base commune
- `Dockerfile.standalone` → image complète permettant d'exécuter directement une application

Différence entre base et standalone

Type d'image	Description	Utilisation
Base	Image de fondation utilisée pour créer d'autres images (ou par des Dockerfiles personnalisés).	Contient uniquement l'environnement d'exécution minimal (ex: Python, Java, GCC, etc.)
Standalone	Image complète capable d'exécuter une application directement.	Utilisée pour les dépôts d'applications "simples" sans Dockerfile personnalisé.

Exemple :

- Une application Python avec un `requirements.txt` → utilise `whanos-python:standalone`.
- Une app qui veut modifier son environnement → crée son propre Dockerfile basé sur `whanos-python:base`.

Langages supportés et spécifications

Langage	Base image	Détection	Compilation	Commande d'exécution
C	`debian:12-slim`	Présence d'un `Makefile`	`make`	`./compiled-app`
Java	`eclipse-temurin:21-jdk-jammy`	`pom.xml`	`mvn package`	`java -jar target/app.jar`
JavaScript	`node:20-slim`	`package.json`	`npm install`	`node .`
Python	`python:3.12-slim`	`requirements.txt`	`pip install -r requirements.txt`	`python -m app`
Befunge	`esolang/befunge93`	`main.bf`	N/A	`befunge app/main.bf`

Fonctionnement du script `build_images.sh`

Ce script automatise la construction de **toutes les images base** (et éventuellement standalone).

Exemple de contenu du script :

```
#!/bin/bash

LANGUAGES=("c" "java" "javascript" "python" "befunge")

for lang in "${LANGUAGES[@]}"; do
    echo "Building base image for $lang..."
    docker build -t whanos-$lang:base -f images/$lang/Dockerfile.base .
    echo "Base image for $lang built!"
    echo
done
```

Utilisation :

```
chmod +x build_images.sh
./build_images.sh
```

Tests locaux (optionnels)

Chaque image peut être testée avec une mini application simple :

Exemple pour Python :

```
test_python_app/
├── app/
│   ├── __init__.py
│   └── __main__.py
└── requirements.txt
```

Contenu de `__main__.py` :

```
print("Hello from my Whanos Python app!")
```

Commande de test :

```
docker build -t whanos-python -f images/python/Dockerfile.standalone
./test_python_app
docker run --rm whanos-python
```

Résultat attendu :

```
 Hello from my Whanos Python app!
```

Intégration avec les autres parties

➤ Jenkins (Personne B)

- Utilisera les **images base** pour builder automatiquement les images d'applications.
- Jenkins déclenchera la création de ces images à chaque push sur un dépôt compatible.

➤ Kubernetes + Ansible (Personne C)

- Se basera sur les images générées pour **déployer les containers** dans le cluster.
- Le fichier `whanos.yml` de chaque projet indiquera le nombre de replicas, ports et ressources à créer.

Conclusion

Cette partie met en place **l’infrastructure Docker de base** de Whanos.
Grâce à ces images, la suite du projet peut :

- Automatiser la création d’images via Jenkins.
- Déployer automatiquement des applications dans Kubernetes.

Récapitulatif

Élément	Statut
Images Base (5 langages)	Créées
Images Standalone (5 langages)	Créées
Script de build automatique	Fonctionnel
Tests unitaires (ex: Python)	Validés
Documentation	Complète

Auteur : Alberic
Rôle : Personne A – Responsable des images Whanos
Statut : ✔️ Terminé – prêt pour intégration Jenkins et déploiement Kubernetes