DevOps & CI/CD Pipeline — TCP/HTTP High-Performance Server

Ce document décrit le pipeline DevOps complet du projet de serveurs TCP/HTTP haute performance (C/POSIX + Python), incluant build C, tests, analyses statiques, sécurité (DevSecOps), benchmarks et déploiement de la documentation.

1. Architecture Globale du Pipeline

Le pipeline est déclenché à chaque push, pull request ou déclenchement manuel. Les jobs principaux couvrent : compilation C, tests unitaires, analyse statique Cppcheck, scan de sécurité CodeQL, benchmarks Python, scan des secrets, analyse des dépendances Python, scan Trivy, génération de la provenance SLSA et déploiement de la documentation sur GitHub Pages.

2. Workflows GitHub Actions

• build.yml : compilation C (make), tests unitaires et exécution de Valgrind basique sur le serveur multi-thread.
• cppcheck.yml : analyse statique de tout le code C du répertoire src/.
• codeql.yml : analyse de sécurité CodeQL sur le code C/C++.
• benchmarks.yml : exécution des benchmarks Python (benchmark_extreme.py ou benchmark.py).
• secrets.yml : détection de secrets via TruffleHog.
• dependency-scan.yml : scan de vulnérabilités des dépendances Python via pip-audit.
• trivy.yml : scan Trivy du système de fichiers pour détecter les vulnérabilités.
• slsa.yml : génération de métadonnées de provenance SLSA pour les artefacts binaires.
• format.yml : vérification du formatage C avec clang-format et lint Markdown.
• deploy_docs.yml : déploiement de la documentation sur GitHub Pages.
• nightly.yml : pipeline complet exécuté chaque nuit (build + test + benchmark).

3. Intégration avec le Projet

Le Makefile du projet fournit les cibles de build et de test : make, make test, make debug, ainsi que des cibles avancées de benchmark et d'UML. Les workflows GitHub Actions appellent ces commandes de manière automatisée, garantissant une reproductibilité des builds et une qualité constante.

4. DevSecOps

Les jobs CodeQL, pip-audit, Trivy et le scan de secrets sont intégrés au pipeline afin de détecter au plus tôt les vulnérabilités potentielles, les dépendances vulnérables ou les fuites de secrets. La provenance SLSA ajoute un niveau de traçabilité sur la chaîne de build et les artefacts produits.

5. Exécution Locale vs CI

En local, le développeur peut exécuter les mêmes étapes via des scripts shell et le Makefile (make clean, make -j, make test, scripts de benchmark Python). En CI, les workflows reproduisent ce comportement sur un environnement Ubuntu propre, ce qui garantit la portabilité du projet ainsi que la détection précoce de régressions.

6. Extensions Futures

Le pipeline est conçu pour être extensible : ajout de tests d’intégration, déploiement vers un environnement de staging, instrumentation de métriques de performance (Prometheus, Grafana, etc.) pour superviser les serveurs en production.

Annexe : Liste des Workflows

- benchmarks.yml
- build.yml
- codeql.yml
- cppcheck.yml
- dependency-scan.yml
- deploy_docs.yml
- format.yml
- nightly.yml
- secrets.yml
- slsa.yml
- trivy.yml