# Déploiement sécurisé sur Kubernetes : Intégration DevSecOps et centralisation des logs

## Vue d'ensemble du projet

Ce projet vise à mettre en place une solution complète de déploiement sécurisé sur Kubernetes avec intégration DevSecOps et centralisation des logs.

## Objectifs principaux

1. **Déploiement Kubernetes** : Créer et déployer une application simple sur Kubernetes
2. **Pipeline de sécurité DevSecOps** : Intégrer des outils de sécurité dans le pipeline CI/CD
3. **Centralisation des logs** : Mettre en place un système de collecte et visualisation des logs

# Analyse comparative des solutions et technologies

## 1. Environnements de déploiement Kubernetes

| Solution | Avantages | Inconvénients | Cas d'usage | Coût |
|---|---|---|---|---|
| **Minikube** | - Installation locale simple<br>- Idéal pour développement<br>- Pas de coût<br>- Support addons intégrés | - Ressources limitées<br>- Un seul nœud<br>- Performance limitée | Développement local<br>Tests de base<br>Apprentissage | Gratuit |
| **Kind** | - Très léger<br>- Démarrage rapide<br>- Supporte multi-nœuds<br>- Intégration CI/CD excellente | - Moins de fonctionnalités<br>- Pas d'interface graphique<br>- Stockage volatile | Tests CI/CD<br>Développement rapide<br>Environnements temporaires | Gratuit |

| Solution | Avantages | Inconvénients | Cas d'usage | Coût |
|---|---|---|---|---|
| **K3s** | - Très léger (moins de 100MB)<br>- Production ready<br>- Facile à installer<br>- Supporte ARM | - Fonctionnalités réduites<br>- Moins d'écosystème<br>- Support communautaire limité | Edge computing<br>IoT<br>Environnements contraints | Gratuit |
| **Docker Desktop** | - Interface utilisateur intuitive<br>- Intégration Docker native<br>- Kubernetes intégré | - Consommation ressources élevée<br>- Licence payante (entreprise)<br>- Limité à un nœud | Développement local<br>Environnements mixtes Docker/K8s | Gratuit (personnel) |

## 2. Gestionnaires de packages Kubernetes

| Solution | Avantages | Inconvénients | Complexité | Écosystème |
|---|---|---|---|---|
| **Helm** | - Standard de facto<br>- Large écosystème de charts<br>- Gestion des versions<br>- Templating puissant | - Courbe d'apprentissage<br>- Complexité pour cas simples<br>- Dépendances multiples | Moyenne | Très large |
| **Kustomize** | - Natif Kubernetes<br>- Approche déclarative<br>- Pas de templating<br>- Simplicité | - Moins de fonctionnalités<br>- Pas de gestion versions<br>- Écosystème limité | Faible | Moyen |
| **YAML brut** | - Simplicité maximale<br>- Contrôle total<br>- Pas de dépendances<br>- Débogage facile | - Duplication de code<br>- Maintenance difficile<br>- Pas de réutilisabilité | Très faible | N/A |

# 3. Outils de sécurité DevSecOps

## Scanners de vulnérabilités

| Outil | Type de scan | Avantages | Inconvénients | Coût | Intégration CI/CD |
|---|---|---|---|---|---|
| **Trivy** | Images, FS, Git | - Très rapide<br>- Base de données complète<br>- Facile à intégrer<br>- Supporte multiples formats | - Uniquement vulnérabilités<br>- Pas d'analyse comportementale | Gratuit | Excellente |
| **Clair** | Images de conteneurs | - Analyse approfondie<br>- API REST<br>- Scalable<br>- Notifications | - Configuration complexe<br>- Ressources importantes<br>- Courbe d'apprentissage | Gratuit | Bonne |
| **Anchore** | Images, conformité | - Analyse de conformité<br>- Politiques personnalisées<br>- Rapports détaillés<br>- Support entreprise | - Version gratuite limitée<br>- Complexité de configuration | Gratuit/Payant | Bonne |

## Analyse de code statique

| Outil | Langages supportés | Avantages | Inconvénients | Coût | Qualité des rapports |
|---|---|---|---|---|---|
| **SonarQube** | 25+ langages | - Analyse complète<br>- Interface web | - Ressources importantes<br>- Configuration | Community/Payant | Excellente |

| Outil | Langages supportés | Avantages | Inconvénients | Coût | Qualité des rapports |
|---|---|---|---|---|---|
| | | riche<br>- Historique des métriques<br>- Règles personnalisables | complexe<br>- Licence payante (fonctionnalités avancées) | | |
| **CodeQL** | 10+ langages | - Analyse sémantique<br>- Requêtes personnalisées<br>- Intégration GitHub<br>- Précision élevée | - Limité aux langages supportés<br>- Courbe d'apprentissage<br>- Ressources importantes | Gratuit (GitHub) | Très bonne |
| **Semgrep** | 20+ langages | - Règles simples<br>- Rapide<br>- Communauté active<br>- CLI intuitive | - Moins de fonctionnalités<br>- Pas d'interface web (version gratuite) | Gratuit/Payant | Bonne |

## Tests de sécurité dynamiques

| Outil | Type de test | Avantages | Inconvénients | Complexité | Automatisation |
|---|---|---|---|---|---|
| **OWASP ZAP** | Web application | - Gratuit et open source<br>- Interface graphique<br>- API complète<br>- Communauté active | - Configuration manuelle<br>- Faux positifs<br>- Ressources importantes | Moyenne | Bonne |
| **Burp Suite** | Web application | - Très précis<br>- Fonctionnalités avancées | - Version gratuite limitée<br>- Coût élevé | Élevée | Moyenne |

| Outil | Type de test | Avantages | Inconvénients | Complexité | Automatisation |
|---|---|---|---|---|---|
| | | - Extensions nombreuses<br>- Support professionnel | (Pro)<br>- Courbe d'apprentissage | | |

## 4. Plateformes CI/CD

| Plateforme | Avantages | Inconvénients | Coût | Écosystème |
|---|---|---|---|---|
| **GitHub Actions** | - Intégration native GitHub<br>- Marketplace d'actions<br>- Gratuit (limites généreuses)<br>- Configuration simple | - Limité aux repositories GitHub<br>- Moins de fonctionnalités avancées<br>- Dépendant de GitHub | Gratuit/Payant | Très large |
| **GitLab CI/CD** | - Intégration complète GitLab<br>- Runners flexibles<br>- DevOps complet<br>- Auto DevOps | - Courbe d'apprentissage<br>- Ressources importantes<br>- Configuration complexe | Gratuit/Payant | Large |
| **Jenkins** | - Très flexible<br>- Plugins nombreux<br>- Contrôle total<br>- Open source | - Maintenance importante<br>- Sécurité à gérer<br>- Interface vieillissante | Gratuit | Très large |
| **Azure DevOps** | - Intégration Microsoft<br>- Outils complets<br>- Scalabilité<br>- Support entreprise | - Coût élevé<br>- Complexité<br>- Vendor lock-in | Payant | Moyen |

# 5. Solutions de centralisation des logs

## Comparaison ELK vs Loki + Grafana

| Critère | ELK Stack | Loki + Grafana | Recommandation |
|---|---|---|---|
| **Complexité d'installation** | Élevée | Faible | Loki pour débuter |
| **Consommation ressources** | Très élevée | Modérée | Loki pour environnements contraints |
| **Capacités de recherche** | Excellentes | Bonnes | ELK pour recherche complexe |
| **Intégration Kubernetes** | Bonne | Excellente | Loki pour Kubernetes |
| **Coût d'infrastructure** | Élevé | Faible | Loki pour budgets limités |
| **Courbe d'apprentissage** | Élevée | Modérée | Loki pour équipes débutantes |
| **Écosystème** | Très mature | En croissance | ELK pour écosystème riche |
| **Performance indexation** | Excellente | Bonne | ELK pour gros volumes |

## Détail des composants

### Stack ELK (Elasticsearch, Logstash, Kibana)

| Composant | Rôle | Avantages | Inconvénients |
|---|---|---|---|
| **Elasticsearch** | Stockage et recherche | - Recherche full-text puissante<br>- Scalabilité horizontale<br>- Agrégations complexes | - Consommation mémoire élevée<br>- Configuration complexe<br>- Coût de stockage |
| **Logstash** | Collecte et transformation | - Nombreux plugins<br>- Transformations complexes<br>- Pipeline flexible | - Consommation ressources<br>- Configuration complexe<br>- Goulot d'étranglement |
| **Kibana** | Visualisation | - Interface riche<br>- Dashboards avancés<br>- Alertes intégrées | - Courbe d'apprentissage<br>- Performances variables<br>- Consommation ressources |

### Stack Loki + Grafana

| Composant | Rôle | Avantages | Inconvénients |
|---|---|---|---|
| **Loki** | Stockage logs | - Très économe en ressources<br>- Indexation par labels<br>- Compatible Prometheus<br>- Version 3.5 récente | - Recherche full-text limitée<br>- Fonctionnalités réduites vs ELK<br>- Moins mature qu'Elasticsearch |
| **Grafana Alloy** | Collecte télémétrie | - Collecteur unifié (logs/métriques/traces)<br>- Remplace Promtail<br>- Configuration moderne<br>- Support OpenTelemetry natif | - Nouveau (courbe d'apprentissage)<br>- Documentation en évolution<br>- Complexité accrue |
| **Promtail** | Collecte logs (legacy) | - Léger et éprouvé<br>- Configuration simple<br>- Autodécouverte Kubernetes | - Uniquement logs<br>- Remplacé par Alloy<br>- Fonctionnalités limitées |
| **Grafana** | Visualisation | - Interface moderne<br>- Dashboards flexibles<br>- Alertes avancées | - Principalement pour métriques<br>- Logs en second plan<br>- Moins de fonctionnalités logs |

**Note importante** : Grafana Alloy est le successeur de Promtail et Grafana Agent, conçu comme collecteur télémétrique unifié.

## Évolution des collecteurs de logs

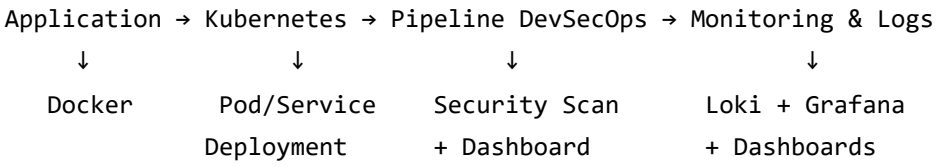
| Génération | Outil | Statut | Capacités | Recommandation |
|---|---|---|---|---|
| **1ère génération** | Promtail | Maintenance | Logs uniquement | Migration vers Alloy recommandée |
| **2ème génération** | Grafana Agent | Déprécié | Logs + Métriques | Migration vers Alloy obligatoire |
| **3ème génération** | **Grafana Alloy** | Actuel | Logs + Métriques + Traces + Profils | **Solution recommandée** |

**Migration Promtail → Alloy** : Grafana Labs recommande de migrer vers Alloy pour bénéficier d'un collecteur unifié et des dernières fonctionnalités.

# 6. Recommandations

| Composant | Recommandation | Justification |
|---|---|---|
| **Kubernetes** | Minikube | **Simplicité et accessibilité :** Idéal pour le développement local, Minikube permet une prise en main rapide de Kubernetes sans les coûts et la complexité d'un cluster cloud. Sa documentation complète en fait un excellent outil d'apprentissage. |
| **Package Manager** | Helm | **Standard de l'industrie et puissance :** Helm est le gestionnaire de paquets de facto pour Kubernetes. Il simplifie la gestion des déploiements complexes grâce à son système de templating et à un vaste écosystème de charts réutilisables. |
| **Scanner vulnérabilités** | Trivy | **Rapidité et intégration facile :** Trivy est reconnu pour sa vitesse d'analyse et sa simplicité d'intégration dans les pipelines CI/CD. Il offre une détection de vulnérabilités complète pour les images de conteneurs, ce qui est essentiel pour une approche DevSecOps. |
| **Analyse code** | SonarQube Community | **Analyse approfondie et suivi qualité :** SonarQube offre une analyse statique complète du code, détectant les bugs, les vulnérabilités et les "code smells". Son interface web permet de suivre l'évolution de la qualité du code de manière centralisée. |
| **CI/CD** | GitHub Actions | **Intégration native et simplicité :** En tant que solution intégrée à GitHub, Actions permet de créer des workflows CI/CD de manière fluide et intuitive. La vaste marketplace d'actions et le généreux plan gratuit en font un choix pragmatique pour ce projet. |
| **Logs** | Loki + Alloy | **Architecture moderne et efficacité :** Cette stack est conçue pour être économique en ressources et nativement intégrée à Kubernetes. Loki indexe uniquement les métadonnées, réduisant les coûts de stockage, tandis que Grafana Alloy est le collecteur de télémétrie unifié de nouvelle génération, assurant une solution d'avenir. |

# Architecture générale




# Plan d'implémentation étape par étape

## Phase 1 : Préparation de l'application

1. **Création de l'application de démonstration**
   - Application web simple
   - Dockerfile pour la containerisation
   - Tests unitaires basiques
2. **Configuration de base**
   - Repository Git avec structure claire
   - README avec instructions de base

## Phase 2 : Déploiement Kubernetes

1. **Fichiers YAML Kubernetes de base**
   - Deployment : définir les pods et réplicas
   - Service : exposer l'application
   - ConfigMap : configuration de l'application
   - Secret : données sensibles (mots de passe, clés)
2. **Déploiement et tests**
   - Déploiement local avec Minikube ou Kind
   - Vérification du fonctionnement
   - Tests de connectivité
3. **Fonctionnalités avancées (bonus)**
   - Helm Chart pour simplifier le déploiement
   - Horizontal Pod Autoscaler (HPA) pour l'autoscaling

## Phase 3 : Pipeline DevSecOps

1. **Configuration CI/CD (GitHub Actions)**
   - Pipeline de build automatique
   - Tests automatisés

- Build et push des images Docker

2. **Intégration des outils de sécurité**
   - **Trivy** : scan de vulnérabilités des images Docker
   - **SonarQube** : analyse de qualité et sécurité du code
   - Configuration simple avec Docker Compose
3. **Dashboard de sécurité**
   - Génération de rapports HTML
   - Intégration dans le pipeline
   - Notifications en cas de problèmes critiques

# Phase 4 : Centralisation des logs

1. **Choix de la stack de logging : Loki + Grafana Alloy**
   - Loki pour le stockage des logs (indexation par labels)
   - Grafana Alloy pour la collecte (successeur de Promtail)
   - Grafana pour la visualisation
2. **Configuration du système de logs**
   - Grafana Alloy pour la collecte des logs (remplace Promtail)
   - Loki pour le stockage et l'indexation
   - Grafana pour la visualisation
3. **Dashboards et monitoring**
   - Dashboard pour les logs d'application
   - Dashboard pour les logs système
   - Alertes sur les erreurs critiques
   - Intégration avec les métriques Prometheus

# Phase 5 : Tests et documentation finale

1. **Tests d'intégration**
   - Test complet du pipeline
   - Vérification des dashboards
   - Test des alertes
2. **Documentation technique**
   - Guide de déploiement
   - Documentation des dashboards
   - Procédures de maintenance

# Choix technologiques simplifiés

## Pour le déploiement Kubernetes

- **Environnement local** : Minikube
- **Package manager** : Helm (optionnel, mais recommandé)
- **Ingress** : NGINX Ingress Controller

## Pour DevSecOps

- **CI/CD** : GitHub Actions (gratuit et simple)
- **Scan sécurité** : Trivy (léger et efficace)
- **Qualité code** : SonarQube Community Edition
- **Dashboard** : Pages HTML simples + GitHub Pages

## Pour les logs

- **Stack choisie** : Loki + Grafana Alloy
- **Collecteur** : Grafana Alloy (successeur de Promtail)
- **Stockage** : Loki avec stockage local
- **Visualisation** : Grafana avec dashboards pré-configurés

# Livrables attendus

## Livrables techniques

1. **Code source**
   - Application de démonstration
   - Fichiers YAML Kubernetes
   - Helm Charts (bonus)
2. **Configuration DevSecOps**
   - Pipelines GitHub Actions
   - Configurations Trivy et SonarQube
   - Dashboards HTML de sécurité
3. **Infrastructure de logging**
   - Configuration Loki + Grafana
   - Dashboards personnalisés
   - Documentation d'utilisation

## Livrables documentaires

1. **Documentation technique**
   - Guide d'installation
   - Guide d'utilisation
   - Architecture détaillée
2. **Rapports**
   - Rapport de sécurité
   - Métriques de performance
   - Recommandations d'amélioration

# Critères de réussite

- ✅ Application déployée et accessible via Kubernetes
- ✅ Pipeline CI/CD fonctionnel avec scans de sécurité
- ✅ Dashboards de sécurité avec rapports HTML
- ✅ Logs centralisés et visualisables dans Grafana
- ✅ Documentation complète et claire
- ✅ Démonstration fonctionnelle du projet complet

# Ressources et outils nécessaires

## Outils de développement

- Docker
- kubectl
- Helm
- Git
- IDE (VS Code)

## Services cloud/locaux

- GitHub (code + CI/CD)
- Minikube (Kubernetes local)
- SonarQube (analyse code)

## Monitoring et visualisation

- Grafana
- Loki + Promtail

- Dashboards HTML personnalisés