KodeMed

Politique de déploiement OpenShift

Pour DevOps, ingénieurs plateforme & architectes IT

Version 2026.5.15.59502 | KodeMed AG

Architecture — Composants

Services déployés dans OpenShift

Service	Technologie	Port	Type
kodemed-server	Java 21, Spring Boot 3.4	8080	Deployment
kodemed-dataserver	Java 21, Spring Boot 3.4	8081	Deployment
kodemed-grouper	Java 21, Spring Boot 3.4	8082	Deployment
kodemed-ui	React 18 / nginx	3000	Deployment
PostgreSQL 16	Externe (pas un pod)	5432	DB externe

Clients Windows (non conteneurisés) :

- KodeMed.Interface — DLL COM .NET 9.0 (intégration HIS)
- KodeMed.CodingClient — App system tray .NET 9.0 (bridge WebSocket)

Limites de ressources par pod

Requêtes/limites CPU et mémoire pour chaque service

Pod	CPU Req	CPU Limit	Mem Req	Mem Limit
kodemed-server	250m	1000m	512Mi	1Gi
kodemed-dataserver	200m	500m	256Mi	1Gi
kodemed-grouper	200m	500m	512Mi	1Gi
kodemed-ui (nginx)	50m	200m	32Mi	64Mi
TOTAL (x2 réplicas)	1400m	4400m	2600Mi	6.1Gi

JVM : -XX:+UseContainerSupport -XX:MaxRAMPercentage=75.0 — respecte automatiquement le memory limit du container.

Haute disponibilité

Minimum 2 réplicas par service avec rolling updates

Service	Réplicas	Rolling Update	Contraintes
kodemed-server	2	maxUnavail: 0, maxSurge: 1	Sticky sessions obligatoires
kodemed-dataserver	2	maxUnavail: 1, maxSurge: 1	Stateless — aucune contrainte
kodemed-grouper	2	maxUnavail: 1, maxSurge: 1	Stateless — aucune contrainte
kodemed-ui	2	maxUnavail: 1, maxSurge: 1	Stateless — aucune contrainte

Sticky Sessions — kodemed-server

- Connexions WebSocket en mémoire JVM (ConcurrentHashMap)
- La Route OpenShift DOIT configurer l'affinité de session :
- haproxy.router.openshift.io/balance: source
- haproxy.router.openshift.io/timeout: 86400s
- Redémarrage pod : clients se reconnectent en 60s — sessions en DB, pas de perte

Health Probes

Sondes readiness, liveness et startup pour tous les services

Service	Endpoint	Port	Init Delay	Période
kodemed-server	/actuator/health	8080	60s	10s
kodemed-dataserver	/actuator/health	8081	60s	10s
kodemed-grouper	/actuator/health	8082	60s	10s
kodemed-ui	/	3000	5s	10s

Types de sondes

- Readiness — détermine si le pod peut recevoir du trafic (retiré du Service si échec)
- Liveness — détermine si le pod est vivant (container redémarré si échec)
- Startup — protège les apps lentes au démarrage (JVM warmup), failureThreshold: 12

Base de données — PostgreSQL externe

Pas de pod DB autorisé dans OpenShift — DB externe avec SSL in-transit

Configuration

- Pas de pod DB dans OpenShift — conforme à la politique du cluster
- SSL in-transit : sslmode=require&sslrootcert=/certs/ca.crt
- Certificat SSL monté via Secret OpenShift (kodemed-db-ssl-cert)
- Connection pool : HikariCP max 15 conn, min 5 idle, par service

Schéma & Migrations

- Schéma partagé : Server préfixe km_app_, DataServer préfixe km_data_
- Migrations Flyway automatiques au démarrage
- Pas de scripts SQL manuels requis
- Toujours sauvegarder la base avant mise à jour

Sécurité des images & SCC

Images immutables, scan de vulnérabilités et Security Context Constraints OpenShift

Images immutables

- readOnlyRootFilesystem: true
- allowPrivilegeEscalation: false
- runAsNonRoot: true, runAsUser: 1001
- Pas d'installation au runtime — config via env vars uniquement
- Pas de nested containers

Scan & Signatures

- Scan Trivy à chaque build CI
- CVE critiques = image bloquée
- Cosign pour intégrité supply-chain
- Base : eclipse-temurin:21-jre-alpine

Image	UID	SCC
kodemed-server	1001	restricted (OK)
kodemed-dataserver	1001	restricted (OK)
kodemed-grouper	1001	restricted (OK)
kodemed-ui	root*	ACTION REQUISE

Action : migrer kodemed-ui vers nginxinc/nginx-unprivileged ou configurer UID > 1000.

Questions ouvertes — Réponses

Réponses aux questions de la réunion

OUI

PDB (PodDisruptionBudget) ?

minAvailable: 1 par service. Avec 2 réplicas, le cluster peut évacuer max 1 pod à la fois.

NON

Opérateur Kubernetes ?

Pas nécessaire. Ressources K8s standard (Deployment, Service, Route) suffisent. Pas de CRD custom.

MIXTE

Stateless vs Stateful ?

DataServer, Grouper, UI = stateless. Server = stateful (WebSocket en JVM). Tous en Deployment. Sticky sessions sur Route.

HELM

Type de déploiement ?

Helm charts recommandés. Paramétrable par env (values-test.yaml, values-prod.yaml). Rollback facile. Compatible ArgoCD/FluxCD.

OUI

Compatible GitOps ?

Images immutables + tags versionnés (YYYY.M.D.BUILD) + config déclarative. Workflow ArgoCD sync.

Matrice RASCI

Responsabilités clairement définies pour chaque acteur

Activité	Mieres IT	Client Plateforme	Client IT
Build & scan images	R	I	I
Publier images sur JFrog	R	S	I
Créer namespace OpenShift	C	R	A
Configurer quotas/limites	C	R	A
Créer Helm charts	R	C	I
Déploiement (test & prod)	S	R	A
Config Routes/Ingress + SSL	C	R	A
Provisionner PostgreSQL + SSL	C	R	A
Gérer Secrets OpenShift	C	R	A
Rolling updates	R (images)	R (deploy)	A
Support applicatif L2/L3	R	S	I
Sauvegarde & restauration DB	C	R	A

R = Responsible | A = Accountable | S = Supportive | C = Consulted | I = Informed

Actions requises

Actions à réaliser avant le déploiement

#	Action	Client Plateforme	Prio
1	Créer le namespace kodemed-test sur OpenShift	Client	Haute
2	Provisionner PostgreSQL 16 avec SSL activé	Client	Haute
3	Configurer le repo Docker dans JFrog	Client	Haute
4	Configurer le repo Helm dans JFrog	Client	Haute
5	Fournir credentials JFrog à Mieres IT	Client	Haute
6	Adapter kodemed-ui pour nginx non-root (SCC)	Mieres IT	Haute
7	Créer les Helm charts	Mieres IT	Haute
8	Créer les Secrets OpenShift (DB, encryption, JWT)	Client	Haute
9	Configurer Routes WebSocket + sticky sessions	Client	Haute
10	Valider la matrice RASCI avec toutes les parties	Client IT	Moyenne