KodeMed

OpenShift-Bereitstellungsrichtlinie

Für DevOps, Platform Engineers & IT-Architekten

Version 2026.5.15.59502 | KodeMed AG

Architektur — Komponenten

In OpenShift bereitgestellte Dienste

Dienst	Technologie	Port	Typ
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	Extern (kein Pod)	5432	Externe DB

Windows-Clients (nicht containerisiert):

- KodeMed.Interface — COM-DLL .NET 9.0 (HIS-Integration)
- KodeMed.CodingClient — System-Tray-App .NET 9.0 (WebSocket-Bridge)

Ressourcenlimits pro Pod

CPU- und Speicher-Requests/Limits für jeden Dienst

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 Replicas)	1400m	4400m	2600Mi	6.1Gi

JVM: -XX:+UseContainerSupport -XX:MaxRAMPercentage=75.0 – respektiert automatisch das Container-Speicherlimit.

Hochverfügbarkeit

Mindestens 2 Replicas pro Dienst mit Rolling Updates

Dienst	Replicas	Rolling Update	Einschränkungen
kodemed-server	2	maxUnavail: 0, maxSurge: 1	Sticky Sessions erforderlich
kodemed-dataserver	2	maxUnavail: 1, maxSurge: 1	Stateless — keine Einschränkungen
kodemed-grouper	2	maxUnavail: 1, maxSurge: 1	Stateless — keine Einschränkungen
kodemed-ui	2	maxUnavail: 1, maxSurge: 1	Stateless — keine Einschränkungen

Sticky Sessions — kodemed-server

- WebSocket-Verbindungen im JVM-Speicher (ConcurrentHashMap)
- OpenShift-Route MUSS Session-Affinität konfigurieren:
- haproxy.router.openshift.io/balance: source
- haproxy.router.openshift.io/timeout: 86400s
- Pod-Neustart: Clients verbinden sich in 60s — Sessions in DB, kein Datenverlust

Health Probes

Readiness-, Liveness- und Startup-Probes für alle Dienste

Dienst	Endpoint	Port	Init Delay	Period
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

Probe-Typen

- Readiness — bestimmt, ob Pod Traffic empfangen kann (aus Service entfernt bei Fehler)
- Liveness — bestimmt, ob Pod lebt (Container wird neugestartet bei Fehler)
- Startup — schützt langsam startende Apps (JVM-Warmup), failureThreshold: 12

Datenbank — Externes PostgreSQL

Keine Datenbank-Pods in OpenShift — externe DB mit SSL in-transit

Konfiguration

- Kein DB-Pod in OpenShift — konform mit Cluster-Richtlinie
- SSL in-transit: sslmode=require&sslrootcert=/certs/ca.crt
- SSL-Zertifikat via OpenShift Secret montiert (kodemed-db-ssl-cert)
- Connection Pool: HikariCP max 15, min 5 idle, pro Dienst

Schema & Migrationen

- Gemeinsames Schema: Server-Präfix km_app_, DataServer-Präfix km_data_
- Flyway-Migrationen laufen automatisch beim Start
- Keine manuellen SQL-Skripte erforderlich
- Datenbank immer vor Upgrade sichern

Image-Sicherheit & SCC

Unveränderliche Images, Schwachstellenscan und OpenShift Security Context Constraints

Unveränderliche Images

- readOnlyRootFilesystem: true
- allowPrivilegeEscalation: false
- runAsNonRoot: true, runAsUser: 1001
- Keine Installationen zur Laufzeit — Konfig nur via Env-Vars
- Keine verschachtelten Container

Scan & Signaturen

- Trivy-Scan bei jedem CI-Build
- Kritische CVEs = Image wird blockiert
- Cosign für Supply-Chain-Integrität
- Basis: 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*	AKTION ERFORDERLICH

Aktion: kodemed-ui auf nginxinc/nginx-unprivileged migrieren oder UID > 1000 konfigurieren.

Offene Fragen — Antworten

Antworten auf die Fragen aus der Besprechung

JA

PDB (PodDisruptionBudget)?

minAvailable: 1 pro Dienst. Mit 2 Replicas kann der Cluster max. 1 Pod gleichzeitig evakuieren.

NEIN

Kubernetes-Operator?

Nicht nötig. Standard-K8s-Ressourcen (Deployment, Service, Route) reichen aus. Keine Custom-CRDs.

GEMISCHT

Stateless vs. Stateful?

DataServer, Grouper, UI = stateless. Server = stateful (WebSocket in JVM). Alle als Deployment. Sticky Sessions auf Route.

HELM

Deployment-Typ?

Helm Charts empfohlen. Parametrisierbar pro Umgebung (values-test.yaml, values-prod.yaml). Einfaches Rollback.

JA

GitOps-kompatibel?

Unveränderliche Images + versionierte Tags (YYYY.M.D.BUILD) + deklarative Konfig. ArgoCD/FluxCD-Workflow.

RASCI-Matrix

Klar definierte Verantwortlichkeiten für jeden Akteur

Aktivität	Mieres IT	Kunde Plattform	Kunde IT
Build & Scan Images	R	I	I
Images in JFrog publizieren	R	S	I
OpenShift-Namespace erstellen	C	R	A
Quotas/Limits konfigurieren	C	R	A
Helm Charts erstellen	R	C	I
Deployment (Test & Prod)	S	R	A
Routes/Ingress + SSL konfig.	C	R	A
PostgreSQL + SSL bereitstellen	C	R	A
OpenShift Secrets verwalten	C	R	A
Rolling Updates	R (Images)	R (Deploy)	A
App-Support L2/L3	R	S	I
DB-Backup & Restore	C	R	A

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

Erforderliche Massnahmen

Aktionspunkte vor dem Deployment

#	Action	Kunde Plattform	Prio
1	Namespace kodemed-test auf OpenShift erstellen	Kunde	Hoch
2	PostgreSQL 16 mit aktiviertem SSL bereitstellen	Kunde	Hoch
3	Docker-Repo in JFrog konfigurieren	Kunde	Hoch
4	Helm-Repo in JFrog konfigurieren	Kunde	Hoch
5	JFrog-Credentials an Mieres IT bereitstellen	Kunde	Hoch
6	kodemed-ui für Non-Root-nginx anpassen (SCC)	Mieres IT	Hoch
7	Helm Charts erstellen	Mieres IT	Hoch
8	OpenShift Secrets erstellen (DB, Encryption, JWT)	Kunde	Hoch
9	Routes WebSocket + Sticky Sessions konfigurieren	Kunde	Hoch
10	RASCI-Matrix mit allen Beteiligten validieren	Kunde IT	Mittel