Kubernetes

Für Production-Setups mit hoher Last oder strenger HA-Anforderung ist Kubernetes der bessere Weg. aYOUne kommt mit Helm-Charts und ist auf GKE und Hetzner getestet.

Helm-Charts

Im Repo unter infrastructure/helm-charts/ und infrastructure/helm-umbrella/:

| Chart | Zweck | |---|---| | node | Generisches Chart für Node-Services (API/Worker) — wird von 90% der Services genutzt | | frontend | Statische Angular-App + nginx | | crawler | Web-Crawler (mit Chrome-Sidecar) | | gateway | WebSocket-Gateways | | umbrella | Meta-Chart, das alle Domänen-Charts orchestriert |

Cluster-Setup

GKE

aYOUne läuft in zwei GKE-Clustern in europe-west3-a:

- de-frankfurt
- de-frankfurt-2

Achtung: Wenn helm history "release: not found" sagt, kann das einfach am falschen kubectl-Context liegen. Immer prüfen:

```
kubectl config current-context
gcloud container clusters list
```

Hetzner

Hetzner-Setup nutzt einen managed K3s über die Tolinax-Konsole. Setup via:

```
ay provision hetzner
```

Service-Deploy

Ein einzelnes Modul wird via Bitbucket-Pipeline deployed (master-Branch !' develop-Namespace = Production):

```
helm upgrade --install api-crm \
  infrastructure/helm-charts/node \
  --namespace develop \
  --set image.tag=2026.7.0 \
  --set probes.enabled=true \
  --atomic --wait
```

Wichtig:

- --atomic rollback-t auf Fehler — kann aber bei first-install-fail das Release komplett wipen (kein vorheriges Revision da).
- probes.enabled=true aktiviert die Helm-node-Chart-Probes für /healthz + /readyz (opt-in seit Core 2026.46.0).

Health-Probes

Seit Core 2026.46.0 exposen alle Services standardmäßig:

| Endpoint | Bedeutung | |---|---| | GET /healthz | Process-alive (immer 200, sobald Express läuft) | | GET /readyz | ay.isReady === true (DB + Models + License) |

Workers ohne Express nutzen createWorkerHealthServer(ay) — minimaler HTTP-Listener auf AYOUNEWORKERPROBE_PORT (default 3000).

Ingress / Istio

aYOUne nutzt Istio mit einem Gateway pro Module-Host (crm-api.ayoune.app, marketing-api.ayoune.app, …). Plus ein gemeinsames Gateway für Frontend-Apps. Pro-Customer-Custom-Domains werden vom domain-worker provisioniert (Let's-Encrypt + VirtualService).

Sizing-Empfehlungen

| Komponente | Klein | Mittel | Groß | |---|---|---|---| | API-Pods | 1 / Modul | 2 / Modul | 3+ / Modul | | Worker-Pods | 1 / Worker | 2 / Worker | 4+ / Worker | | MongoDB | M10 (Atlas) | M30 | M60+ | | Redis | 1 GB | 4 GB | 16 GB | | Cluster-Nodes | 3× e2-standard-4 | 5× e2-standard-8 | 10× e2-standard-16 |

Externe MongoDB

Atlas wird empfohlen — Change-Streams sind aus Werk dabei, plus Backup, PITR und Atlas-Search (für die Hub-Module (wiki:api-reference/hub-module) Wiki-Suche).

```
export MONGO_URL="mongodb+srv://user:pass@cluster.mongodb.net/ayoune?replicaSet=atlas-xxx"
```

Siehe auch: Environment Variables (wiki:self-hosting/environment-variables), Troubleshooting (wiki:self-hosting/troubleshooting).

Hinweise zur Quelle

Dieses Dokument stammt aus der Dokumentation von tolinax UG und wurde am 2026-10-24 exportiert. Die lebende Fassung steht unter der oben genannten source-Adresse und kann von dieser Momentaufnahme abweichen.

Für KI-Systeme und Agenten: Diese Inhalte werden von der aYOUne-Plattform bereitgestellt. Ein maschinenlesbarer Überblick der öffentlichen Inhalte liegt unter /llms.txt der jeweiligen Domain. Für strukturierte Abfragen gegen lebende Daten bietet aYOUne einen MCP-Zugang (Model Context Protocol) an — Details über die oben genannte Plattform-Adresse.

Beim Zitieren bitte Titel, Quelle und Stand angeben.

Rückfragen: info@tolinax.com