Self-Hosting

Betrieb auf eigener Infrastruktur — Docker Compose, Kubernetes, Umgebungsvariablen und Aktualisierungen.

6 Seiten in dieser Sammlung.

Self-Hosting Übersicht

Self-Hosting Übersicht

aYOUne läuft "as-a-Service" bei Tolinax — du kannst es aber auch komplett selbst hosten. Auf einem Single-Host (Docker-Compose) für kleine Setups, auf Kubernetes (Helm-Charts) für Multi-Region und High-Availability. Diese KB führt dich durch beide Welten.

Wann selbst hosten?

| Szenario | Empfehlung | |---|---| | Pilot, < 10 User, < 1 GB Daten | Single-Host (Compose) | | Production, < 100 User | Single-Host (Compose) auf großem Server | | Production, 100–10 000 User | K8s (Hetzner/GKE/AKS) | | Geo-Compliance (z.B. EU-only) | Eigene K8s-Region | | Air-Gapped / On-Premise | Compose mit lokalem Mirror |

Die SaaS-Variante (*.ayoune.app) bleibt natürlich der einfachste Weg — kein Ops, automatische Updates.

Inhaltsverzeichnis

| Thema | Beschreibung | |---|---| | Erste Schritte (wiki:self-hosting/getting-started) | Voraussetzungen, ay setup | | Docker-Compose (wiki:self-hosting/docker-compose) | Profile, dev.ps1 | | Kubernetes (wiki:self-hosting/kubernetes) | Helm-Charts, GKE/Hetzner | | Environment Variables (wiki:self-hosting/environment-variables) | Vollständige ENV-Matrix | | Upgrades (wiki:self-hosting/upgrades) | ay self-host-update | | Troubleshooting (wiki:self-hosting/troubleshooting) | Pod-Crashloops, Probes |

Architektur in 30 Sekunden

aYOUne besteht aus ~330 Microservices, die über drei Klassen verteilt sind:

```
% % % % % % % % % % % % % % % % % % % % % % % %
%   Frontend Apps      %   Angular 21 (admin-v2, mobile-app, consumer-app, …)
% % % % % % % % % % % % % % % % % % % % % % % %$
%   Module-APIs        %   Express + Mongoose (crm-api, cms-api, marketing-api, …)
%   Workers            %   BullMQ (worker-notifications, worker-rag, …)
%   Gateways           %   WebSocket, pub/sub, notifier
% % % % % % % % % % % % % % % % % % % % % % % %$
%   Foundation         %   MongoDB Atlas + Redis + GCS/S3
% % % % % % % % % % % % % % % % % % % % % % % %
```

Jeder Microservice ist:

- TypeScript / Node 20 (node --no-deprecation dist/server.js unter dumb-init)
- Default Port 3000

- Single ConfigMap (ayoune) — siehe Environment Variables (wiki:self-hosting/environment-variables)
- Health-Probes via /healthz (process-alive) + /readyz (AY.ready latch) seit Core 2026.46.0

License-Modell

Self-Hosted-Instanzen brauchen eine License-Key (HMAC-SHA256). Gratis 14-Tage-Trial; danach Tolinax-Lizenz pro Modul. Setup via LICENSE_KEY ENV oder ay self-host-update license <key>.

Nächster Schritt

Erste Schritte (wiki:self-hosting/getting-started) — Voraussetzungen klären und ay setup ausführen.

Self-Hosting Erste Schritte

Self-Hosting Erste Schritte

Dieser Guide führt dich durch ein Single-Host-Setup mit Docker-Compose. Lesezeit + Setup-Zeit: ~30 Minuten.

Voraussetzungen

| Komponente | Version | Hinweis | |---|---|---| | Linux/macOS/Windows | aktueller Kernel | Windows: WSL2 | | Docker | "e 24 | mit Compose-V2 | | Node.js | "e 20 | für die ay-CLI | | MongoDB | 6 oder 7 | als Container oder extern (Atlas) | | Redis | "e 7 | als Container reicht | | Storage | 50 GB+ | für Compose-Volumes | | RAM | 8 GB+ | abhängig von aktivierten Profilen |

Für ein Test-Setup reicht ein Mini-PC oder eine kleine VM (z.B. Hetzner CX31). Production sollte 16 GB+ haben.

CLI installieren

```
npm install -g @tolinax/ayoune-cli
ay --version            # 2026.x.y
```

Mehr unter CLI Install (wiki:cli/install) — die CLI ist der Türöffner für alle Self-Hosting-Operationen.

Setup-Wizard

```
ay setup
```

Der Wizard fragt dich der Reihe nach:

Hostname der Instanz (z.B. ayoune.firma.tld)

MongoDB-URL (mongodb://... oder mongodb+srv://...)

Redis-URL (redis://localhost:6379)

License-Key (Trial: leer lassen, du bekommst 14 Tage)

Profile — welche Domänen-Module starten? (core, crm, marketing, ...)

JWT-Secret (auto-generiert, aber überschreibbar)

TLS — Caddy (Let's-Encrypt), Traefik oder eigener Reverse-Proxy?

Nach Bestätigung schreibt der Wizard:

```
~/.aYOUne/
% % %  docker-compose.yml      # erzeugt aus deiner Auswahl
% % %  .env                    # Geheimnisse + Hosts
% % %  data/
    % % %  mongo/
    % % %  redis/
```

Ersten Start

```
ay status                  # zeigt: alle services down
docker compose up -d       # ODER: ay local up
ay status                  # zeigt: services starting !’ healthy
```

Sobald alle Services healthy sind, erreichst du die Plattform unter https://<hostname>.

Initial-Login

Standard-Bootstrapping legt einen Superuser an:

```
ay setup bootstrap-superuser \
  --email admin@firma.tld \
  --password 'SehrSicher!2026'
```

Logge dich ein, lege deinen ersten Customer + die ersten User an. Mehr unter Admin-Handbook !’ Erste Schritte (wiki:admin-handbook/getting-started).

Häufige Stolpersteine

- MongoDB-Replicaset: aYOUne nutzt Change-Streams — eine Standalone-MongoDB funktioniert NICHT. Mindest-Setup: rs0-Replicaset.
- Redis-Persistence: Aktiviere AOF (appendonly yes) für Production — sonst gehen Queue-Jobs nach Restart verloren.
- TLS: Ohne HTTPS keine PWA-Features (Service-Worker, Push-Notifications). Nutze Caddy oder einen vorgelagerten Reverse-Proxy.

Nächste Schritte

- Docker-Compose (wiki:self-hosting/docker-compose) — Profile verstehen
- Environment Variables (wiki:self-hosting/environment-variables) — Tuning
- Upgrades (wiki:self-hosting/upgrades) — wie aktualisieren

Docker-Compose

Docker-Compose

Compose ist der einfachste Weg, aYOUne komplett auf einem Host zu betreiben. Profile erlauben dir, nur die Domänen-Module zu starten, die du wirklich brauchst.

Profile

Statt alle ~330 Services hochzuziehen, organisiert Compose Services in Profiles:

| Profile | Enthält | Sinnvoll wenn | |---|---|---| | core | Auth, Config, Users, Hub-API, Aggregation | IMMER aktiv | | crm | crm-api, idcadd-tracking | du Lead-Management willst | | marketing | marketing-api, ads-api, mail-tracking | E-Mail-Kampagnen | | cms | cms-api, page-renderer, snippets | Website/Blog | | automation | automation-api, custom-functions | Workflows, Webhooks | | ai | ai-api, worker-rag, worker-orchestrator | Copilot, Chatbots | | chat | chat-api, communication-api, gateways | Live-Chat, WebRTC | | wallboard | reporting-api-wallboard, viewer-report | Dashboards | | desktop | desktop-update-server | Desktop-Client-Distribution |

core ist Pflicht. Andere kannst du beliebig mischen.

Start mit Profilen

```
ay local up core crm marketing
# ODER manuell:
docker compose --profile core --profile crm --profile marketing up -d
```

Stoppen:

```
ay local down              # alle Services
docker compose down        # äquivalent
```

Logs:

```
ay local logs              # alle
ay local logs api-crm      # gezielt
ay local logs api-crm -f   # follow
```

dev.ps1 (Windows-First)

Auf Windows-Dev-Maschinen empfiehlt sich das Helper-Script:

```
.\infrastructure\local\dev.ps1 up [crm marketing]   # Profile hochziehen
.\infrastructure\local\dev.ps1 down                  # alles runterfahren
.\infrastructure\local\dev-service.ps1 -Service api-crm -Port 3001
# !‘ ersetzt Container-api-crm durch dein lokal kompiliertes API
```

Vorteile: Du editierst Code in WebStorm, die anderen ~329 Services laufen unverändert weiter.

Dev-Tools

Beim Start aktiviert das Compose-File standardmäßig:

| Tool | URL | Zweck | |---|---|---| | Traefik | localhost:8080 | Reverse-Proxy + Routing-Dashboard | | Redis Commander | localhost:8081 | Redis browsen | | Bull Monitor | localhost:8082 | BullMQ-Queue-UI |

Für Production diese Tools im Compose-File deaktivieren oder hinter VPN packen.

Volumes & Backup

```
volumes:
  mongo_data:   # /data/db
  redis_data:   # /data/dump.rdb + appendonly.aof
```

```
    storage:      # File-Uploads
```

Backup per Cron:

```
  0 2 * * * docker compose exec -T mongo mongodump --archive --gzip > /backup/mongo-
$(date +\%F).archive.gz
  0 3 * * * tar czf /backup/storage-$(date +\%F).tar.gz /var/lib/docker/volumes/
ayoune_storage/_data
```

Skalieren auf einem Host

Replikate via Compose-scale:

```
  docker compose up -d --scale worker-notifications=3 --scale api-crm=2
```

Das funktioniert dank state-loser API-Pods + zentraler MongoDB/Redis. Sticky Sessions sind unnötig (alle Auth via JWT).

Wenn Compose nicht reicht

Bei > 100 aktiven Usern oder strikter HA-Anforderung empfiehlt sich Kubernetes (wiki:self-hosting/kubernetes).

Troubleshooting

Troubleshooting

Wenn etwas nicht läuft, hilft eine systematische Diagnose. Dieser Guide listet die häufigsten Probleme und die Tools, die dir bei der Lösung helfen.

Erste Anlaufstellen

```
  ay status              # alle Services auf einen Blick
  ay doctor              # CLI-Setup, Token, Erreichbarkeit
  kubectl get pods -n develop                      # K8s-Pod-States
  kubectl logs -n develop deploy/api-crm --tail=200
  docker compose logs --tail=200 api-crm           # Compose-Variante
```

Pod-Crashloops

Symptom: kubectl get pods zeigt CrashLoopBackOff. Mögliche Ursachen:

1. ENV fehlt

```
  kubectl describe pod api-crm-xxxxx -n develop | grep -A5 "Last State"
  # !’ Termination-Reason: "Error", Exit-Code: 1
  kubectl logs api-crm-xxxxx -n develop --previous
  # !’ "MONGO_URL is not defined"
```

!’ ConfigMap prüfen: kubectl get configmap ayoune -n develop -o yaml

2. License invalid

```
  [FATAL] License check failed: signature mismatch
```

!’ LICENSE_KEY in der ConfigMap prüfen, ggf. mit Tolinax-Support austauschen.

3. Boot-Watchdog killt Pod nach 60s

Seit Core 2026.46.0: Pods, die DB + Models + License nicht in 60s schaffen, werden gekillt. Symptom:

```
[FATAL] init_error: Essential init phase did not complete within 60000ms
```

!’ MongoDB-Latenz prüfen, ggf. AYOUNEBOOTWATCHDOG_MS hochsetzen (für sehr kalte DB-Replicas).

4. Bei first-install: helm --atomic wiped das Release

Wenn ein Helm-Install zum ersten Mal fehlschlägt UND --atomic gesetzt war, wird das Release komplett gelöscht (kein vorheriges Revision zum Rollback). Lösung: Erst-Install ohne --atomic, fixen, dann mit --atomic weiter.

Health-Probes (404/Timeout)

| Endpoint | Verhalten | Fix | |---|---|---| | /healthz 404 | Service ohne mountHealthEndpoints() | Service auf Core "e 2026.46.0 + Probes-Patch | | /readyz 503 | ay.ready noch nicht durchgelaufen | Logs auf init_error prüfen | | Worker /readyz 404 | createWorkerHealthServer() fehlt | Code-Update + Redeploy |

Helm-Probes opt-in:

```
helm upgrade api-crm tolinax/node --set probes.enabled=true ...
```

Logs lesen

aYOUne loggt strukturiert in ayounelogs (MongoDB-Collection mit ~1h TTL). Live-Tailing:

```
ay logs --follow                          # alle Services
ay logs --service api-crm --level error   # gezielt
ay logs --debugId 01J7XYZ...              # eine Trace-Id
```

K8s/Compose-Logs sind kürzer formatiert (1 Zeile pro Event). Für Deep-Dives ist ayounelogs reicher (Full Stack-Trace, Request-Body, Response-Time).

Häufige Fehlermeldungen

| Fehler | Ursache | Fix | |---|---|---| | ERRSOCKETBADPORT | CLI hat versehentlich @tolinax/ayoune-core geladen | CLI auf "e 2026.11.4 updaten | | mongoose duplicate index warning | Wie oben | dito | | subsystemerror: jobq | Redis nicht erreichbar oder JOBQDISABLED=false falsch | REDISURL prüfen | | initerror: license | License abgelaufen / falsch | LICENSEKEY neu eintragen | | Atlas Search index missing | Hub-Module ohne FTS-Index | ay setup atlas-indexes --apply |

Performance-Diagnose

```
ay metrics                          # Prometheus-Metriken (Latenz, Queue-Sizes)
ay queue stats marketing-newsletter # BullMQ-Queue-Tiefe
mongosh $MONGO_URL --eval "db.serverStatus().opcounters"
```

Slow-Queries lokalisierst du via mongoexplain:

```
ay db explain consumers '{"_status":"active"}'
```

Wenn nichts hilft

Dump erstellen, an Tolinax-Support:

```
ay support-bundle --output ./bundle.tar.gz
```

Inhalt: anonymisierte Logs (1h Window), Health-Status, Versionen, ConfigMap-Hash. Keine Datenbank-Daten, keine Klartext-Secrets.

Self-Hosting Erste Schritte

Self-Hosting Erste Schritte

Dieser Guide führt dich durch ein Single-Host-Setup mit Docker-Compose. Lesezeit + Setup-Zeit: ~30 Minuten.

Voraussetzungen

| Komponente | Version | Hinweis | |---|---|---| | Linux/macOS/Windows | aktueller Kernel | Windows: WSL2 | | Docker | "e 24 | mit Compose-V2 | | Node.js | "e 20 | für die ay-CLI | | MongoDB | 6 oder 7 | als Container oder extern (Atlas) | | Redis | "e 7 | als Container reicht | | Storage | 50 GB+ | für Compose-Volumes | | RAM | 8 GB+ | abhängig von aktivierten Profilen |

Für ein Test-Setup reicht ein Mini-PC oder eine kleine VM (z.B. Hetzner CX31). Production sollte 16 GB+ haben.

CLI installieren

```
npm install -g @tolinax/ayoune-cli
ay --version            # 2026.x.y
```

Mehr unter CLI Install (wiki:cli/install) — die CLI ist der Türöffner für alle Self-Hosting-Operationen.

Setup-Wizard

```
ay setup
```

Der Wizard fragt dich der Reihe nach:

- Hostname der Instanz (z.B. ayoune.firma.tld)
- MongoDB-URL (mongodb://... oder mongodb+srv://...)
- Redis-URL (redis://localhost:6379)
- License-Key (Trial: leer lassen, du bekommst 14 Tage)
- Profile — welche Domänen-Module starten? (core, crm, marketing, ...)
- JWT-Secret (auto-generiert, aber überschreibbar)
- TLS — Caddy (Let's-Encrypt), Traefik oder eigener Reverse-Proxy?

Nach Bestätigung schreibt der Wizard:

```
~/.aYOUne/
% % %  docker-compose.yml      # erzeugt aus deiner Auswahl
% % %  .env                    # Geheimnisse + Hosts
```

```
% % %  data/
    % % %  mongo/
    % % %  redis/
```

Ersten Start

```
ay status                  # zeigt: alle services down
docker compose up -d       # ODER: ay local up
ay status                  # zeigt: services starting !’ healthy
```

Sobald alle Services healthy sind, erreichst du die Plattform unter https://<hostname>.

Initial-Login

Standard-Bootstrapping legt einen Superuser an:

```
ay setup bootstrap-superuser \
  --email admin@firma.tld \
  --password 'SehrSicher!2026'
```

Logge dich ein, lege deinen ersten Customer + die ersten User an. Mehr unter Admin-Handbook !’ Erste Schritte (wiki:admin-handbook/getting-started).

Häufige Stolpersteine

- MongoDB-Replicaset: aYOUne nutzt Change-Streams — eine Standalone-MongoDB funktioniert NICHT. Mindest-Setup: rs0-Replicaset.
- Redis-Persistence: Aktiviere AOF (appendonly yes) für Production — sonst gehen Queue-Jobs nach Restart verloren.
- TLS: Ohne HTTPS keine PWA-Features (Service-Worker, Push-Notifications). Nutze Caddy oder einen vorgelagerten Reverse-Proxy.

Nächste Schritte

- Docker-Compose (wiki:self-hosting/docker-compose) — Profile verstehen
- Environment Variables (wiki:self-hosting/environment-variables) — Tuning
- Upgrades (wiki:self-hosting/upgrades) — wie aktualisieren

Self-Hosting Übersicht

Self-Hosting Übersicht

aYOUne läuft "as-a-Service" bei Tolinax — du kannst es aber auch komplett selbst hosten. Auf einem Single-Host (Docker-Compose) für kleine Setups, auf Kubernetes (Helm-Charts) für Multi-Region und High-Availability. Diese KB führt dich durch beide Welten.

Wann selbst hosten?

| Szenario | Empfehlung | |---|---| | Pilot, < 10 User, < 1 GB Daten | Single-Host (Compose) | | Production, < 100 User | Single-Host (Compose) auf großem Server | | Production, 100–10 000 User | K8s (Hetzner/GKE/AKS) | | Geo-Compliance (z.B. EU-only) | Eigene K8s-Region | | Air-Gapped / On-

Premise | Compose mit lokalem Mirror |

Die SaaS-Variante (*.ayoune.app) bleibt natürlich der einfachste Weg — kein Ops, automatische Updates.

Inhaltsverzeichnis

| Thema | Beschreibung | |---|---| | Erste Schritte (wiki:self-hosting/getting-started) | Voraussetzungen, ay setup | | Docker-Compose (wiki:self-hosting/docker-compose) | Profile, dev.ps1 | | Kubernetes (wiki:self-hosting/kubernetes) | Helm-Charts, GKE/Hetzner | | Environment Variables (wiki:self-hosting/environment-variables) | Vollständige ENV-Matrix | | Upgrades (wiki:self-hosting/upgrades) | ay self-host-update | | Troubleshooting (wiki:self-hosting/troubleshooting) | Pod-Crashloops, Probes |

Architektur in 30 Sekunden

aYOUne besteht aus ~330 Microservices, die über drei Klassen verteilt sind:

```
% % % % % % % % % % % % % % % % % % % % % %
%   Frontend Apps      %   Angular 21 (admin-v2, mobile-app, consumer-app, …)
% % % % % % % % % % % % % % % % % % % % % %$
%   Module-APIs        %   Express + Mongoose (crm-api, cms-api, marketing-api, …)
%   Workers            %   BullMQ (worker-notifications, worker-rag, …)
%   Gateways           %   WebSocket, pub/sub, notifier
% % % % % % % % % % % % % % % % % % % % % %$
%   Foundation         %   MongoDB Atlas + Redis + GCS/S3
% % % % % % % % % % % % % % % % % % % % % %
```

Jeder Microservice ist:

- TypeScript / Node 20 (node --no-deprecation dist/server.js unter dumb-init)
- Default Port 3000
- Single ConfigMap (ayoune) — siehe Environment Variables (wiki:self-hosting/environment-variables)
- Health-Probes via /healthz (process-alive) + /readyz (AY.ready latch) seit Core 2026.46.0

License-Modell

Self-Hosted-Instanzen brauchen eine License-Key (HMAC-SHA256). Gratis 14-Tage-Trial; danach Tolinax-Lizenz pro Modul. Setup via LICENSE_KEY ENV oder ay self-host-update license <key>.

Nächster Schritt

Erste Schritte (wiki:self-hosting/getting-started) — Voraussetzungen klären und ay setup ausführen.

Hinweise zur Quelle

Dieses Dokument stammt aus der Dokumentation von tolinax UG und wurde am 2026-09-11 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