HIVEGRID

INFRASTRUKTUR-DOKUMENTATION

Server-Struktur & Hetzner-Estate

Komplette Bestandsaufnahme: Hosts, Netzwerk, Dienste, Deploy

6 Hosts | Hetzner (EU) | Coolify Control-Plane | WireGuard-Mesh | Zentrale Dienste

Zweck: Onboarding- & Betriebsreferenz für die gesamte Server-Landschaft.

Quelle der Wahrheit: Repo `infra/` (README, STANDARDS, ADRs, Runbooks).

Stand: 14. Juni 2026

Modell: per-Produkt getrennte Workloads, zentral gesteuert über Coolify, zentrale Hilfsdienste über ein privates WireGuard-Mesh.

HIVEGRID

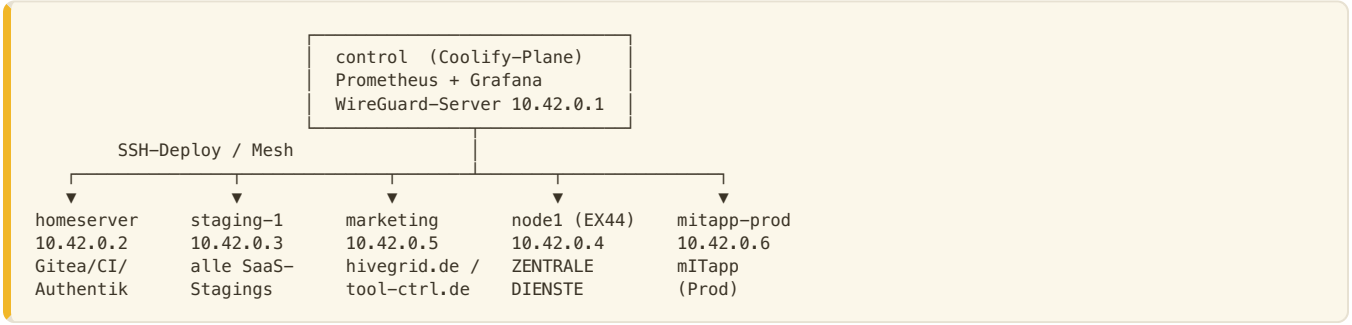
Inhalt

1. Überblick & Architekturmodell
2. Server-Inventar (alle Hosts)
3. Hetzner-Projekt-Struktur
4. Netzwerk & WireGuard-Mesh
5. Orchestrierung (Coolify, Traefik)
6. Zentrale Dienste (node1)
7. Observability & Logging
8. Aufteilung pro Produkt
9. Deploy-Kette
10. DNS & Domains
11. Secrets & Zugänge
12. Backups & Disaster Recovery
13. Standards & Konventionen
14. Offene Punkte

1. Überblick & Architekturmodell

Schlankes, per-Produkt getrenntes SaaS-Infra-Modell mit zentraler Steuerung – alles auf Hetzner (EU).

- **Compute:** Mehrere Produkt-Stagings teilen sich einen Host (`staging-1`); Produktion bekommt pro Produkt ein eigenes Hetzner-Projekt/VM (ADR-009).
- **Control-Plane:** Eine zentrale Coolify-Instanz (`control`) verwaltet alle Hosts und Deployments.
- **Hilfsdienste:** Einmal zentral auf dem dedizierten `node1` (Sentry, Loki, Meilisearch, ClamAV, mCaptcha, NSFW-API), erreichbar über ein privates WireGuard-Mesh.
- **CI/CD:** Build-once in der Gitea-Registry, Deploy über Coolify (Staging automatisch, Prod manuell/Promote).
- **Governance:** `infra/` -Repo mit STANDARDS, nummerierten ADRs und Runbooks ist die Quelle der Wahrheit.



2. Server-Inventar

Host	Provider / Typ	Spec	Public-IP	Mesh-IP	Rolle
`control`	Hetzner Cloud · cx33	4C / 8 GB / 80 GB	195.201.133.26	10.42.0.1	Coolify-Control-Plane, Prometheus + Grafana, WireGuard-Server
`staging-1`	Hetzner Cloud · cpx42	8C / 15 GB / 75 GB	46.225.5.93	10.42.0.3	Gemeinsamer SaaS-Staging-Host (boilerplate, mITapp, künftig HiveGrid)
`node1` (EX44)	Hetzner Dedicated (Robot)	20C / 62 GB / 436 GB	176.9.111.179	10.42.0.4	Zentrale Dienste: Sentry, Loki, Meilisearch, ClamAV, mCaptcha, NSFW-API
`marketing`	Hetzner Cloud · cx33	4C / 8 GB / 80 GB	128.140.109.153	10.42.0.5	Marketing-Sites (hivegrid.de, tool-ctrl.de)
`mitapp-prod`	Hetzner Cloud · cpx22	2C / 4 GB / 80 GB	178.104.213.13	10.42.0.6	mITapp Produktion
`homeserver`	Self-hosted (DDNS)	16C / 30 GB / 935 GB	DDNS	10.42.0.2	Gitea, CI-Runner, Authentik, Vaultwarden, interne Tools

Alle Cloud-Hosts in Hetzner-Regionen nbg1/fsn1 (Deutschland). `node1` im RZ FSN1-DC8. SSH per Schlüssel (`~/.ssh/node1` für alle Hetzner-Hosts).

3. Hetzner-Projekt-Struktur

Workloads sind in getrennte Hetzner-Cloud-Projekte aufgeteilt (Abrechnung/Isolation). Produktion erhält pro Produkt ein eigenes Projekt (ADR-009).

Projekt	Zweck	VM(s)	Status
DevOps	Coolify-Control-Plane	`control` (cx33)	live
Staging	SaaS-Staging-Pool	`staging-1` (cpx42)	live
Marketing	Marketing-Workload	`marketing` (cx33)	live
probono	mITapp-Produktion	`mitapp-prod` (cpx22)	live
HiveGrid	HiveGrid-Cloud-Prod (geplant, ADR-009)	—	geplant
Boilerplate	Boilerplate-Prod (geplant, ADR-009)	—	geplant

Dazu: 1 Dedicated-Server (`node1` , Hetzner Robot) für zentrale Dienste und der self-hosted `homeserver` für CI/SSO.

4. Netzwerk & WireGuard-Mesh

Alle Hosts hängen in einem privaten WireGuard-Mesh `10.42.0.0/24` mit Stern-Topologie: `control` ist Server (UDP 51820), alle anderen sind Peers. Inter-Host-Verkehr (Metrics-Scrapes, zentrale Dienste) läuft ausschließlich über Mesh-IPs – nicht über öffentliche Endpunkte.

Host	Interface	Mesh-IP	Rolle im Mesh
control	wg0	10.42.0.1/24	Server
homeserver	wg1	10.42.0.2/24	Peer (eigenes wg0 bleibt Heim-VPN)
staging-1	wg0	10.42.0.3/24	Peer
node1	wg1	10.42.0.4/24	Peer (eigenes wg0 separates VPN)
marketing	wg0	10.42.0.5/24	Peer
mitapp-prod	wg0	10.42.0.6/24	Peer

5. Orchestrierung (Coolify, Traefik)

Coolify v4 auf `control` (Dashboard `https://saas-ctrl.com`) ist die zentrale Steuerung. Jeder Ziel-Host ist als Server registriert; pro Produkt gibt es ein Projekt, pro Stufe ein Environment, pro Dienst eine Resource.

- **Resource-Modus `dockerimage` :** Coolify baut nicht selbst, sondern zieht vorgefertigte Images aus der Gitea-Registry.
- **Reverse-Proxy:** Traefik (`coolify-proxy`) je Host, TLS via Let's Encrypt (ACME). Auf `node1` hat Traefik seit der ADR-013-Migration das frühere Host-Caddy ersetzt.
- **Konvention:** Projekt = Produkt · Environment = Stufe (staging/production) · Resource = Dienst.

6. Zentrale Dienste (node1)

Einmal zentral betrieben, von allen Produkten über das Mesh genutzt. Index-/Key-Konventionen pro Produkt sorgen für Trennung.

Dienst	Erreichbar	Zweck / Hinweise
Sentry	`sentry.saas-ctrl.com`	Error-Tracking. 1 Org, 1 Team pro Produkt, 1 Projekt pro Codebasis; Staging/Prod über Environment-Tags.
Loki	`10.42.0.4:3100` (mesh)	Zentrale Logs; Promtail-Collector auf jedem Host; 7 Tage Retention.
Meilisearch	`10.42.0.4:7700` (mesh)	Volltextsuche; pro Produkt ein Scoped-Key, Index-Präfix `<produkt>-*` .
ClamAV	`10.42.0.4:3310` (mesh)	Virenscan per TCP-INSTREAM; Default fail-closed für Uploads.
NSFW-API	`10.42.0.4:3000` (mesh)	Bild-Klassifikation; Default fail-open.
mCaptcha	`captcha.saas-ctrl.com`	Captcha-Service (Sitekey-verwaltet).

7. Observability & Logging

Zentral auf `control` : **Prometheus** scrapt jeden Host über das Mesh (alle 30 s), **Grafana** (`grafana.saas-ctrl.com`) visualisiert. Logs zentral in **Loki**, Errors in **Sentry**.

- **Scrape-Targets je Host:** `node-exporter` (:9100) und `docker-stats-exporter` (:9487) auf der jeweiligen Mesh-IP.
- **Labels:** `project` , `environment` , `server` , `service` – damit lassen sich Dashboards pro Produkt/Host filtern.
- **Dashboards:** `estate-overview` (flottenweit) plus je Host/Produkt eines (`project-staging` , `project-hetzner-ex44` , `project-mitapp` , …).
- **Offen:** AlertManager-/Alert-Routing noch nicht aktiv; Dashboards aktuell nicht versioniert.

8. Aufteilung pro Produkt

HiveGrid Cloud (SaaS) Prod geplant

SaaS-App in aktiver Entwicklung (Git-Downstream des Boilerplates). Noch nicht deployt. Folgt bei Reife dem Estate-Modell: Staging auf `staging-1` (oder eigenes Projekt), Prod als eigenes Hetzner-Projekt „HiveGrid" gemäß ADR-009; zentrale Dienste auf `node1` . Nicht verwechseln mit der Marketing-Site `hivegrid.de` .

HiveGrid Marketing-Site live

Statische/Next.js-Site auf `marketing` , Domain `hivegrid.de` (Cloudflare-proxied), TLS über Traefik. Leads lokal in SQLite-Volume.

Boilerplate SaaS Staging live

Staging auf `staging-1` (Coolify-Projekt „boilerplate" / „validation"): web + api + 5 Worker, Postgres (Coolify-managed) + PgBouncer, Authentik unter `auth.boiler.saas-ctrl.com` ; Suche/Virenscan/Errors zentral auf `node1` . **Prod** laut ADR-009 als eigenes Projekt geplant, noch nicht live.

mITapp live

Prod auf `mitapp-prod` (cpx22): App + Worker (Image-Tag `:prod`), Uploads-Sidecar (nginx), Postgres 16 + Redis 7 (Coolify-managed). Domains `mitapp.de` (kanonisch) + `api.mitapp.de` (mobil) sowie Legacy `mitapp.mitup.de` (muss `/api/*` + `/.well-known/*` für ausgelieferte Mobile-Apps bedienen). **Staging** auf `staging-1` (`staging.mitapp.de` , Basic-Auth). CI/CD nach Modell „Build-once / Promote".

9. Deploy-Kette

```
git push origin main
   │
   ▼  Gitea Actions (Runner auf homeserver, Label "toolctrl-ci")
   │  build image  →  push  git.tool-ctrl.de//:latest + :sha-
   │
   ▼  Coolify (control)  –  Resource-Modus "dockerimage"
   │  pull pre-built image  →  per SSH auf Ziel-Host  →  Container-Rolling-Restart (~25-30s)
   │
   ▼  Traefik (Ziel-Host)  →  TLS-Termination (Let's Encrypt)
   │
   ▼  Cloudflare DNS (+ Proxy bei user-facing Domains)  →  live
```

- **Staging:** automatisch bei Push auf `main` (Tag `:latest` + `:sha-…`).
- **Produktion (Modell „Promote"):** manueller `promote.yml`-Dispatch retaggt ein bereits gebautes SHA-Image auf `:prod` (kein Rebuild) und deployt es. Rollback = altes SHA neu deployen.
- **Registry-Auth:** pro Host `docker login` gegen die Gitea-Registry.

10. DNS & Domains (Cloudflare)

DNS zentral in Cloudflare. User-facing-Domains laufen über den Cloudflare-Proxy; Origins, die selbst Let's-Encrypt-Zertifikate ausstellen, stehen auf „DNS-only". Transaktionsmails über Mailjet (DKIM/SPF je Zone), eingehende Mail über mailbox.org.

Domain / Subdomain	Ziel-Host	Zweck
`saas-ctrl.com`	control	Coolify-Dashboard
`grafana.saas-ctrl.com`	control	Monitoring
`sentry.saas-ctrl.com` · `captcha.saas-ctrl.com`	node1	Zentrale Dienste
`boiler.saas-ctrl.com` · `auth.boiler…`	staging-1	Boilerplate-Staging + Admin-SSO
`hivegrid.de`	marketing	HiveGrid-Marketing-Site
`tool-ctrl.de` (+ `git`, `auth`,…)	marketing / homeserver	Marketing + interne Tools (DDNS)
`mitapp.de` · `api.mitapp.de` · `staging.mitapp.de`	mitapp-prod / staging-1	mITapp Prod/API/Staging

Aktive Zonen u. a.: `saas-ctrl.com`, `hivegrid.de`, `mitapp.de`, `mitup.de`, `tool-ctrl.de`, `power-linked.de` (reserviert). TLS-Modus durchgehend „Strict".

11. Secrets & Zugänge

- **App- & Service-Secrets:** zentral über **Coolify** verwaltet – als Environment-Variablen je Resource, nie im Repo.
- **Admin-/CLI-Zugänge** (Hetzner-Cloud-API, Coolify-API, Cloudflare) liegen lokal unter `~/.cli-tokens/` (chmod 600).
- **Inter-Host:** Coolify→Host per dediziertem SSH-Key; Apps→zentrale Dienste über das private Mesh (mesh-only).
- **Admin-SSO & Passwortverwaltung:** Authentik + Vaultwarden auf dem `homeserver`.

Grundsatz aus STANDARDS: keine Secrets im Repo, keine Klartext-Weitergabe in Chats. (Konkrete Schlüsselwerte und Dateipfade sind in diesem Dokument bewusst ausgespart.)

12. Backups & Disaster Recovery

> **Status: in Arbeit**
> Eine vollständige, automatisierte Backup-Strategie ist noch nicht abgeschlossen – wichtig, bevor eine DB zum alleinigen Speicher von Kundendaten wird.

Komponente	Disposabel?	Backup / Recovery
Coolify-State (`/data/coolify` auf control)	nein (alle Deploy-Configs)	nur lokaler Teil-Snapshot; automatisiertes S3-Export geplant
Produkt-DBs (Postgres, Prod/Staging)	nein (Kundendaten)	Coolify-Auto-Backup zu aktivieren/dokumentieren
Sentry / Loki / ClamAV / mCaptcha (node1)	ja (nur Historie/Sig-DB)	kein Backup – werden bei Bedarf neu aufgesetzt
Meilisearch-Index	teilweise	aus Postgres reindizierbar; S3-Snapshots geplant

Geplant: Synology-NAS zieht per Hyper Backup von `control` und `homeserver`; Off-Site-/Objekt-Speicher über **Hetzner Object Storage (S3, EU)**.

Recovery-Kurzform: `node1`-Ausfall → Dienste neu deployen (Daten verworfen). `control`-Ausfall → `/data/coolify` wiederherstellen, Coolify starten, Apps neu deployen. Produkt-DB → aus Postgres-Backup wiederherstellen (sobald aktiviert).

13. Standards & Konventionen

Verbindlich aus `infra/STANDARDS.md` (Kurzfassung):

1. Errors → zentrale Sentry (1 Team/Produkt, 1 Projekt/Codebasis, Environment-Tags).
2. Jeder Host liefert Metrics (node-exporter + docker-stats) und Logs (Promtail → Loki).
3. Jedes SaaS bekommt ein Grafana-Dashboard (gefiltert nach `project` / `server`).
4. Volltextsuche über zentrale Meilisearch (Index-Präfix, Scoped-Key).
5. Admin-SSO über zentrale Authentik.
6. Repos in Gitea, privat per Default; Dockerfile Pflicht.
7. Coolify ist das Deploy-Ziel: Projekt = Produkt, Environment = Stufe, Resource = Dienst.
8. Domains in Cloudflare (proxied für user-facing, DNS-only für LE-Origins).
9. Secrets außerhalb des Repos (`~/.cli-tokens/` , chmod 600, keine Chat-Pastes).
10. Backup dokumentieren, bevor etwas „live" geht.
11. ADR für nicht-triviale Entscheidungen (`decisions/` , nummeriert).
12. Uploads: Virenscan über zentrale ClamAV (fail-closed), Bildprüfung über NSFW-API (fail-open).

14. Offene Punkte

Neutral zusammengefasst aus README-/LESSONS-To-dos (Stand Juni 2026):

- **Release:** erster mlTapp-`:prod`-Promote noch ausstehend; Legacy-Compose-Stack nach Bewährungsfenster abbauen.
- **Prod-Topologie:** eigene Hetzner-Projekte für Boilerplate- und HiveGrid-Cloud-Prod gemäß ADR-009 anlegen.
- **Backups:** Coolify-State-Export, Produkt-DB-Backups und NAS-/Off-Site-Ziel finalisieren.
- **Sicherheit:** Secret-Rotation, 2FA auf Cloudflare, Hetzner-Cloud-Firewall (SSH auf bekannte IPs), Cloudflare-Token auf „scoped" reduzieren.
- **Observability:** Alert-Routing aktivieren, Grafana-Dashboards versionieren.

Stand 14. Juni 2026. Quelle der Wahrheit ist das `infra/`-Repository (`README.md` , `STANDARDS.md` , `LESSONS.md` , `decisions/` , `runbooks/`). Bei Abweichungen gilt das Repo. Konkrete Geheimnisse/Schlüssel sind hier bewusst ausgespart.