

ILIAS KONFERENZ 2026 · HAMBURG

Von Tagen auf Stunden

ILIAS Orchestrierung in der Praxis

René Goebbels

Technischer Service Owner · Heinrich-Heine-Universität



Die administrative Realität: Schmerzpunkte im klassischen Betrieb

Warum imperative, VM-basierte Setups an ihre Grenzen stoßen



01

Der Faktor Zeit & manuelle Prozesse

- Imperative Installationen und Konfigurationen fressen Tage statt Stunden.
- Hoher personeller Aufwand bei der Bereitstellung neuer Umgebungen.

→ Tage statt Stunden



02

Betriebsrisiko & Wissenssilos

- **Configuration Drift:** schleichend voneinander abweichende Stände der VMs.
- Fehlende Standardisierung führt zu aufwendiger Dokumentation und Wissenssilos.

→ Configuration Drift



03

Strategischer Druck & Abhängigkeiten

- Steigende Anforderungen an Performance (z. B. E-Prüfungen) und Compliance.
- **Die Lock-in-Falle:** wachsende Abhängigkeit von proprietären Großkonzernen.
- **Ziel:** digitale Souveränität durch konsequent offene Standards.

→ Vendor Lock-in

Strategische Pfeiler der neuen Betriebsarchitektur

Die Antwort auf zeitfressende Prozesse, Betriebsrisiken und Abhängigkeiten



01

Deklaratives Prinzip (IaC & CaC)

- **Soll-Zustand im Code:** Systemzustände statt manuell geklickt beschrieben.
- **Auf Knopfdruck:** vollautomatisierte Aufsetzprozesse ersetzen Tage an Vorarbeit.
- **Entlastung:** Fokus verschiebt sich von Administration zu Weiterentwicklung.

→ Zielzustand als Code



02

Reproduzierbarkeit

- **Schluss mit Drift:** Dev-, Test- und Prod-Umgebungen sind zu 100 % identisch.
- **Git als Single Source of Truth:** jede Änderung historisiert und nachvollziehbar.
- **Keine Wissenssilos:** Dokumentation entsteht automatisch durch den Code.

→ Single Source of Truth



03

Digitale Souveränität

- **Offene Standards:** Container, Kubernetes, Helm schützen vor Vendor Lock-in.
- **Selbstbestimmung:** unabhängig von proprietären Cloud-Anbietern und ihren Diktaten.
- **Volle Kontrolle:** auditierbare Infrastruktur nach eigenen Compliance-Vorgaben.

→ Kein Vendor Lock-in

Von der Code-Zeile zur skalierbaren Instanz

Der automatisierte End-to-End-Prozess der Cloud-Native-Architektur

git push → live in Minuten



INPUT

Git & Code-Deklaration

Infrastruktur-, Helm- & ILIAS-Konfiguration als Code — **Single Source of Truth**. Jede Änderung per Merge Request & Peer Review versioniert.



DEPLOY

GitOps-Sync via Flux

Flux gleicht Ist- mit Soll-Zustand ab — **keine manuellen Klicks** oder SSH-Zugriffe auf dem Cluster nötig.



BUILD & VALIDATE

CI/CD & Security-Gates

Pipelines bauen gehärtete OCI-Images. **Security by Design:** CVE-Scans & automatische SBOM-Erstellung.



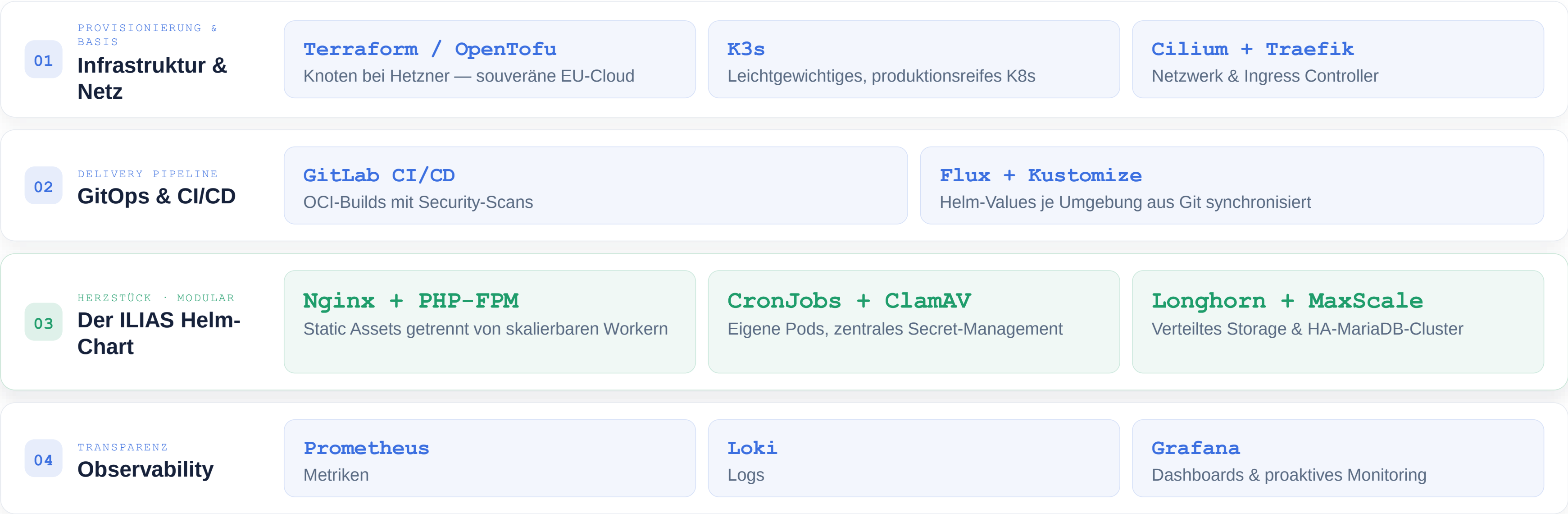
RUN & SCALE

Kubernetes Runtime

Entkoppelte Laufzeit (PHP-Worker, DB-Cluster, Storage) **skalierbar** für Lastspitzen.

Die technische Zielarchitektur im Überblick

Entkoppelter, souveräner Cloud-Native-Stack auf Basis offener Standards



Der Chart ist Anbieter Unabhängig — es kann auf “jedem” Provider oder In-House eingesetzt werden.

Die echte Hürde: Transformation von Mensch & Organisation

Warum Infrastructure as Code vor allem ein Kulturwandel ist

KLASSISCH · IMPERATIV	→	GITOPS-MINDSET · DEKLARATIV
 Ticket für neue VM stellen	→	Merge Request im Git 
 Warten auf Firewall-Freigabe	→	Pipeline provisioniert automatisch 
 Server händisch einrichten	→	Team schaut auf denselben Code 



Die Hürde beim IaC-Provisioning

- **Berechtigungs- & Freigabekultur:** Wer darf Ressourcen (CPU, RAM, Netze) automatisiert anfordern?
- **Schnittstellen-Reibung:** von manuellen Ticket-Systemen zu automatisierten Pipelines.
- **Vertrauens-Dilemma:** Abgabe manueller „Kontrolle“ an die CI/CD- & GitOps-Engine.



Der Kulturwandel im Betriebsteam

- **Vom Administrator zum Platform Engineer:** Shift-Left bei Verantwortung & Security.
- **Abbau von Wissenssilos:** Konfiguration ist im Code für das ganze Team einsehbar.
- **Fehlerkultur & Vertrauen:** Fehler werden im Code gefixt & versioniert statt live „geflickt“.

Technik ≈ Wochen, Kulturwandel = der eigentliche Change — und die zwingende Voraussetzung, um die Früchte der Automatisierung zu ernten.

Härtetest & Messbare Ergebnisse

Validierte Belastbarkeit und konkreter Effizienzgewinn im Praxistest

PRAXISTEST & BELASTBARKEIT

1.000 parallele
Prüflinge

SZENARIO · E-PRÜFUNGEN

Gezielte Skalierung

- Auslegung der Architektur auf **1.000 parallele Prüflinge**.
- Kontrollierte Kapazitätserhöhung der Worker-Pods **direkt deklarativ über das Helm-Chart**.

0 tagelange
Wartungsfenster

LAUFENDER BETRIEB

Nahtloser Update-Betrieb

- **Minimale Downtimes** bei Releases durch lückenlose Automatisierung aller Migrations- & Deployment-Schritte.
- **Schnelle Wiederherstellbarkeit** (Resilienz) über Helm Charts & Backups.

VORHER-NACHHER-EFFIZIENZ



Bereitstellungszeit: **von Tagen auf Stunden**

Instanzen werden per Knopfdruck rein deklarativ ausgerollt.



Reproduzierbarkeit: **100 % auditierbar**

Git als Single Source of Truth eliminiert Configuration Drift & Wissenssilos.



Compliance & Security

Security by Design durch automatisierte CVE-Scans & SBOM in der GitLab-Pipeline.



Team-Entlastung

Abbau von Wissenssilos und Fokusverschiebung hin zu strategischem Platform Engineering.

Fazit & Impulse für das ILIAS-Ökosystem

Cloud-Native als strategische Zukunftschance für den ILIAS-Betrieb



CORE TAKEAWAY

Krisenfest & zukunftssicher

Cloud-Native, GitOps & IaC sind kein Hype, sondern ein **Fundament für verlässlichen, skalierbaren ILIAS-Betrieb**.



IMPULS 1 · CORE

Configuration-as-Code

Deklarative Schnittstellen für Systemeinstellungen — **weg von manuellen Admin-UI-Klicks & fragilen DB-Dumps**.



IMPULS 2 · ARCHITEKTUR

Datenbank-Skalierung

Evolution der ILIAS-DB-Architektur für **native, hochverfügbare & dynamisch skalierbare Cluster**.



CALL TO ACTION

Lassen Sie uns gemeinsam Cloud-Native-Standards im ILIAS-Ökosystem etablieren!