Infrastruktur-Spezifikation

Semesterprojekt – DevOps und Cloud Computing 3B

Benjamin Feichtlbauer (Teamleiter), Kevin Forter, Mikulovic Luka, Tepsurkaev Abdulah

Bachelor Informatik / 3. Semester

Lehrveranstaltung: DevOps und Cloud Computing 3B

20. Oktober 2025

Einleitung

In diesem Semesterprojekt planen wir die Infrastruktur für eine DevOps-basierte CI/CD-Umgebung in der Amazon-Cloud. Die Infrastruktur soll ein privates Netzwerk mit DNS-Auflösung, Versionsverwaltung, Continuous Integration, Authentifizierung via LDAP sowie die Möglichkeit zur Ausführung von Pipelines bereitstellen. Als Beispielanwendung wählen wir ein einfaches LaTeX-Projekt: ein hochgeladenes `.tex`-Dokument soll automatisch per GitLab-Pipeline kompiliert und als PDF über eine *saubere URL* bereitgestellt werden. Dafür nutzt der GitLab-Runner NGINX zur Dateiauslieferung, und der GitLab-Server veröffentlicht die PDFs per Reverse-Proxy unter `https://gitlab.<domain>/pdf/...`. Ziel dieser Spezifikation (Milestone 1) ist das Gesamtdesign: Topologie, Dienste, Security-Regeln, Tests und Rollen.

Infrastrukturplanung

Im Folgenden werden die für Milestone 1 geforderten Aspekte der Infrastruktur spezifiziert. Für die grafische Darstellung der Topologie verwenden wir Platzhalter; die konkreten Visualisierungen werden im Laufe des Projekts erstellt.

Netzwerktopologie

Die Projektumgebung besteht aus einer einzelnen Amazon VPC mit einem **öffentlichen Subnetz** (IPv4-CIDR `10.0.0.0/24`). Alle Server erhalten feste private IP-Adressen; nur der GitLab-Server besitzt eine Elastic IP für den externen Zugriff. Interne Namensauflösung erfolgt über die Zone `ci.intern`, die auf zwei DNS-Servern (primary / secondary) repliziert wird.

Die Architektur ist bewusst minimal gehalten: GitLab dient als zentrales Gateway und Reverse-Proxy, der Runner hostet NGINX zur Ablage der erzeugten PDFs, und die DNS-Server stellen interne Erreichbarkeit sicher. Ein optionaler LDAP-Server wird in Milestone 3 ergänzt.

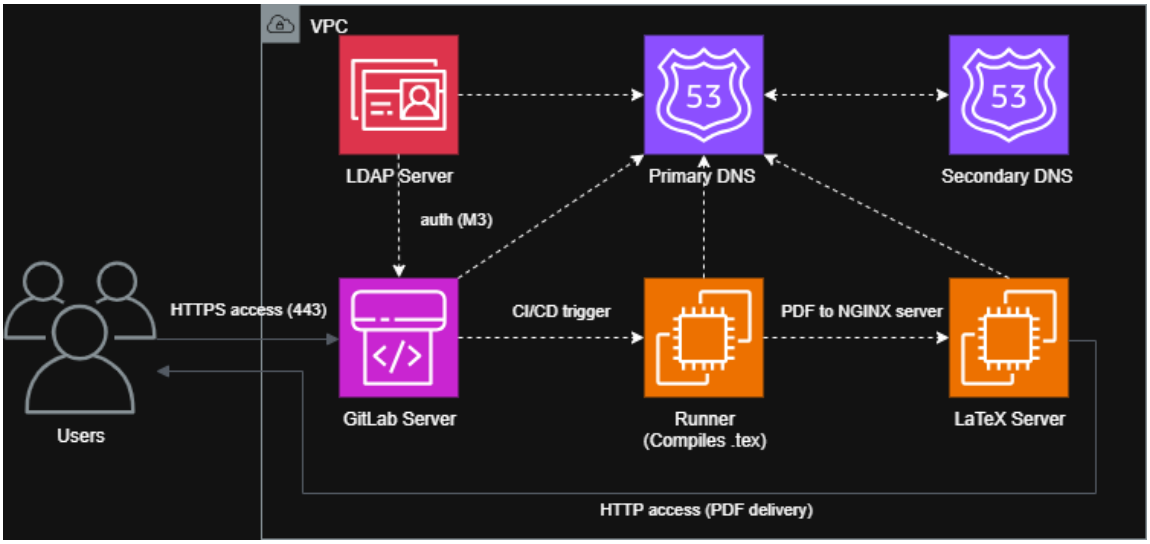

Abbildung 1: Geplante Netzwerktopologie der CI/CD-Umgebung

Interne Adressen und Namen:

- **primary.dns.ci.intern** – 10.0.0.10 | DNS (Zonenverwaltung)
- **secondary.dns.ci.intern** – 10.0.0.11 | DNS-Replikat / Failover
- **gitlab.ci.intern** – 10.0.0.20 | GitLab Server + Reverse-Proxy (TLS)
- **runner.ci.intern** – 10.0.0.21 | GitLab Runner + NGINX (PDF-Hosting)

Reverse-Proxy-Pfad: GitLab veröffentlicht die unter `/var/www/pdfs/{dev,latest}` auf dem Runner liegenden PDFs per Reverse-Proxy als `https://gitlab.<domain>/pdf/...`.

Sicherheitsgruppen

Für jedes System wird eine eigene AWS-Security Group angelegt. Die Inbound-Regeln sind strikt minimal; Outbound ist standardmäßig erlaubt.

- **primary-dns-sg**: UDP/TCP 53 von *runner-sg* und *gitlab-sg*; SSH (TCP 22) nur von Team-VPN/IP-Bereich.
- **secondary-dns-sg**: wie primary; zusätzlich Zonentransfer (IXFR/AXFR) von/zu 10.0.0.10.
- **gitlab-sg**: HTTPS (TCP 443) aus dem Internet, HTTP (TCP 80) nur Redirect; DNS (UDP/TCP 53) intern; SSH (TCP 22) nur Admins/VPN.
- **runner-sg**: HTTP (TCP 80) *nur* aus *gitlab-sg* (Reverse-Proxy); SSH (TCP 22) nur intern/VPN; DNS (53) intern. *Keine* öffentliche Erreichbarkeit.

Services, Betriebssysteme und Pakete

- **DNS-Server (primary/secondary)**: Ubuntu 22.04 LTS, `bind9`, `bind9utils`; Zone `ci.intern` (Forward/Reverse), Zonentransfer per IXFR/AXFR (kein manuelles rsync für Zonen!).
- **GitLab-Server**: Ubuntu 22.04 LTS, *GitLab CE (Omnibus)*, Let's-Encrypt-Zertifikat; optionale Elastic IP (stabile öffentliche Erreichbarkeit).
- **GitLab Runner**: Ubuntu 22.04 LTS, *gitlab-runner* (Docker-Executor), Docker Engine (20.10+); zusätzlich *NGINX* zur Auslieferung der PDFs aus `/var/www/pdfs/`.
- **LDAP-Server (Milestone 3)**: OpenLDAP auf Ubuntu 22.04 LTS; Integration als Auth-Quelle für GitLab (Bind+Search); LDAPS optional/später.

Geplante Tests

DNS

- Forward-Lookup: `dig gitlab.ci.intern @primary.dns.ci.intern` → interne IP.
- Reverse-Lookup: `dig -x 10.0.0.20 @primary.dns.ci.intern` → PTR korrekt.

- Forwarder: `dig google.com @primary.dns.ci.intern` → Antwort vorhanden.
- Zonentransfer: `dig AXFR ci.intern @secondary.dns.ci.intern` → Erfolg.
- Failover: primary stoppen → `dig gitlab.ci.intern @secondary` liefert weiter Antworten.

GitLab Server

- Externer Zugriff: `https://gitlab.<domain>` (TLS gültig).
- Repo anlegen, clone, push; Projekt sichtbar.
- Runner-Registrierung sichtbar (grüner Status im GitLab-UI).

Runner + NGINX (PDF-Bereitstellung)

- CI-Job kompiliert LaTeX (`pdflatex` zweimal, non-stop-mode).
- Deploy-Job: Kopie der PDFs nach `/var/www/pdfs/{dev,latest}` via SSH/rsync.
- NGINX-Check (intern): `curl -I http://runner.ci.intern/pdf/latest/doc.pdf` → `200 OK`.
- Reverse-Proxy-Check (extern): `curl -I https://gitlab.<domain>/pdf/latest/doc.pdf` → `200 OK`, `Content-Type: application/pdf`.

Security Groups

- Externes `nmap` auf GitLab-IP: nur 443 offen (80 Redirect).
- Interne Scans: runner nur von gitlab auf 80 erreichbar; dns nur intern.

LDAP (Milestone 3)

- `ldapsearch -x -H ldap://ldap.ci.intern -b "dc=ci,dc=intern"` liefert Test-User.
- GitLab-Login via LDAP-User möglich; Logs zeigen Bind/Search auf `ldap.ci.intern`.

Teamrollen und Verantwortlichkeiten

- **Benjamin Feichtlbauer (Teamleiter)**: Projektkoordination, CI/CD-Pipeline, NGINX-Ablage + Reverse-Proxy-Konzept, Kosten-& Risk-Abschätzung.
- **Kevin Forter**: DNS (primary/secondary, Zone/Reverse, Transfer), Security-Groups, Unterstützung GitLab-Server (TLS/Omnibus).
- **Mikulovic Luka**: Runner-Setup (Docker-Executor), NGINX-Konfiguration, SSH-Deploy-Pfad `/var/www/pdfs/{dev,latest}`, Hardening.
- **Tepsurkaev Abdulah**: Schritt-für-Schritt-Dokumentation, Testfälle/Checklisten, Vorbereitung LDAP-Planung (Schema, User, Bind/Search).

Kostenabschätzung (vorläufig)

Vier kleine Instanzen (`t3.micro`) + eine Elastic IP (nur GitLab) + EBS (20–30 GB/Instanz) und geringer Traffic bleiben im Rahmen der 50 USD AWS-Academy-Credits, wenn Instanzen bei Nichtnutzung gestoppt werden. Detaillierte Kalkulation (AWS Pricing Calculator) folgt in Milestone 3.

Fazit

Die vorliegende Infrastruktur-Spezifikation legt das Fundament für den Aufbau einer funktionsfähigen DevOps-Umgebung. Durch die kosteneffiziente Topologie (eine VPC, ein Subnetz), klare Security-Groups sowie die Konzentrierung der PDF-Auslieferung auf den Runner mit Veröffentlichung über den GitLab-Reverse-Proxy erreichen wir ein professionelles, aber budgetschonendes Setup. In den nächsten Meilensteinen setzen wir die Planung auf AWS um, verifizieren DNS/GitLab/Runner, etablieren die CI/CD-Stages und ergänzen LDAP; anschließend dokumentieren wir die Schritte und demonstrieren die Funktion in Video und Präsentation.