Overleaf

Overleaf Community Edition – Installationsanleitung für einen abgeschotteten (Air-Gapped) Behördenserver

Zielbild: Ein Server, der für die Dauer der Installation temporär Internetzugang über einen Proxy erhält, alle benötigten Komponenten herunterlädt, danach vollständig vom Netz getrennt wird und ohne weitere Internetverbindung dauerhaft betrieben werden kann.

Rahmenbedingungen dieses Vorhabens (Entscheidungen aus der Planung):

- Kein SSO/AD-Anbindung nötig → lokale Nutzerverwaltung, Accounts manuell per Admin angelegt, Einladung per Mail-Link.
- Kein Server Pro (Kostengrund) → **offizielle Overleaf Community Edition**, kein Fremd-Fork.
- Kein Bedarf an Sandboxed Compiles (kleine, bekannte Nutzergruppe: Studierende, die ihre Bachelorarbeiten schreiben) — als bewusst akzeptiertes Restrisiko dokumentiert, nicht übersehen.
- BibTeX **und** Biber müssen funktionieren (biblatex-Workflows).
- Installation über ein **manuell geschriebenes** `docker-compose.yml`, nicht über das Overleaf Toolkit — direkter nachvollziehbar, weniger bewegliche Teile, entspricht dem hier als Referenz laufenden Aufbau (nur ohne dessen Fork und OIDC-Anbindung).
- Kein Docker-freier Weg möglich: Overleaf CE ist architektonisch ~10 einzelne Microservices, die nur im offiziellen Container-Image lauffähig zusammengeschnürt sind; ein natives Setup ist von Overleaf selbst nicht unterstützt.

---

1. Architekturüberblick

Drei Docker-Container, über ein einziges `docker-compose.yml` orchestriert:

Container	Image	Zweck
`sharelatex`	`sharelatex/sharelatex` (offizielles CE-Image)	Web-App, Compiler-Orchestrierung
`mongo`	`mongo:8.0` (o. kompatible Version)	Projektdaten, Nutzer, Metadaten — läuft als Single-Node Replica-Set (`rs0`), das ist für Overleaf-Transaktionen zwingend, auch bei einer einzelnen Instanz
`redis`	`redis:6.2`	Sessions, Queues

Dazu ein separat gemountetes **TeX-Live-Verzeichnis** (nicht im Image, sondern als eigenes Volume), damit Version und Paketumfang unabhängig vom Docker-Image kontrolliert werden können — das ist auch der Grund, warum ein zusätzlicher Schritt für Biber nötig ist (siehe Abschnitt 6).

---

2. Voraussetzungen an den Server

- Linux-Server (Debian/Ubuntu empfohlen), Docker Engine + Docker Compose Plugin installiert.
- Ausreichend Plattenplatz: TeX Live (Vollschema) ca. 9–10 GB, Docker-Images ca. 2–3 GB, Projekt-/DB-Daten je nach Nutzerzahl gering (im Referenzsystem mit wenigen Projekten < 200 MB).
- Temporärer, geplanter Internetzugang über einen Proxy **nur für die Installationsphase**.
- Admin-Rechte, um systemd-Units anzulegen (Docker-Proxy-Konfiguration) und ggf. ein internes TLS-Root-Zertifikat in den System-Trust-Store einzuspielen.

---

3. Phase 1 – Vorbereitung (online, mit Proxyfreigaben)

3.1 Docker für den Behördenproxy konfigurieren

Der Docker-Daemon liest keine normalen Shell-Umgebungsvariablen. Proxy-Konfiguration erfolgt über ein systemd-Drop-in:

```
sudo mkdir -p /etc/systemd/system/docker.service.d
sudo tee /etc/systemd/system/docker.service.d/http-proxy.conf <<'EOF'
[Service]
Environment="HTTP_PROXY=http://proxy.eure-behoerde.de:PORT"
Environment="HTTPS_PROXY=http://proxy.eure-behoerde.de:PORT"
Environment="NO_PROXY=localhost,127.0.0.1"
EOF

sudo systemctl daemon-reload
sudo systemctl restart docker
```

Ohne diesen Schritt schlägt jeder `docker pull` mit Timeout fehl.

3.2 TLS-Interception beachten

Falls der Proxy TLS aufbricht (eigene CA, MITM-Prinzip, wie in vielen Behördennetzen üblich): Das interne Root-Zertifikat **muss** auf dem Server im System-Trust-Store landen (z. B. `/usr/local/share/ca-certificates/` + `update-ca-certificates` unter Debian/Ubuntu), sonst brechen sowohl `docker pull` als auch der TeX-Live-Installer mit TLS-Fehlern ab. Das ist die häufigste Fehlerquelle bei Downloads über Behörden-Proxies — vorab mit der Netzwerk-/Security-Abteilung klären.

3.3 Vollständige Liste der benötigten Proxyfreigaben

#	Zweck	Host(s)	Protokoll	Nötig für
1	Docker-Image-Metadaten/Auth	`auth.docker.io`	HTTPS/443	alle drei Container-Images
2	Docker-Registry-API	`registry-1.docker.io`	HTTPS/443	alle drei Container-Images

#	Zweck	Host(s)	Protokoll	Nötig für
3	Docker-Image-Layer-CDN	`production.cloudflare.docker.com`	HTTPS/443	eigentliche Bilddaten (großer Traffic-Anteil)
4	(optional, Redirect-Ziel)	`index.docker.io`, `docker.io` , `hub.docker.io`	HTTPS/443	Fallback, falls (1)–(3) nicht reichen
5	TeX Live	**ein einzelner, fest gewählter CTAN-Mirror**, z. B. `ftp.tu-chemnitz.de` (Liste: https://ctan.org/mirrors/mirmon)	HTTPS/443	vollständige TeX-Live-Installation inkl. BibTeX/Biber

Da kein Toolkit von GitHub geklont wird, entfällt die GitHub-Freigabe — es müssen nur Docker Hub und ein einzelner CTAN-Mirror erreichbar sein. Das ist die kleinstmögliche Freigabeliste für dieses Vorhaben.

Bewusst NICHT freigeben: `mirror.ctan.org` als Ziel selbst — das ist ein GeoIP-Redirector, der bei jedem Aufruf auf einen von ~50 wechselnden Drittservern verweisen kann. Für eine nachvollziehbare, eng gefasste Firewall-Freigabe stattdessen direkt einen festen Mirror aus der Mirror-Liste wählen und diesen sowohl für den Bootstrap-Download als auch als `-repository`-Parameter des Installers verwenden (siehe 3.5).

3.4 Docker-Images ziehen

```
docker pull sharelatex/sharelatex
docker pull mongo:8.0
docker pull redis:6.2
```

3.5 TeX Live frisch herunterladen (voller Umfang, fester Mirror)

```
# Bootstrap-Installer von GENAU EINEM festen Mirror holen
wget https://ftp.tu-chemnitz.de/pub/tex/systems/texlive/tlnet/install-tl-unx.tar.gz
tar xzf install-tl-unx.tar.gz
cd install-tl-*

# Installer auf denselben Mirror festnageln (keine weiteren Redirects)
# scheme-full = komplette Distribution inkl. BibTeX, Biber, biblatex, alle .bst-Stile
./install-tl -repository https://ftp.tu-chemnitz.de/pub/tex/systems/texlive/tlnet \
```

```
            -scheme full
```

Warum `scheme-full` statt einer schlanken Auswahl: BibTeX ist zwar in jedem Schema enthalten (`collection-basic`), Biber inkl. `biblatex` steckt aber in `collection-bibtexextra`. Im Air-Gap-Betrieb lässt sich später nichts mehr unkompliziert nachinstallieren — deshalb lieber einmalig alles ziehen (~9–10 GB) als hinterher festzustellen, dass ein Stil/Paket fehlt.

GPG-Verifizierung der heruntergeladenen Pakete lässt der Installer standardmäßig aktiv — **nicht** mit `-no-verify-downloads` deaktivieren, damit die Herkunft der Pakete nachvollziehbar bleibt (Supply-Chain-Aspekt, gerade im Behördenkontext relevant).

Ergebnis liegt typischerweise unter `/usr/local/texlive/<JAHR>/` — dieses Verzeichnis wird im nächsten Schritt als Volume in den Container gemountet.

---

4. Phase 2 – Biber-Setup-Skript anlegen

`bibtex` funktioniert nach der `scheme-full`-Installation ohne weiteres Zutun. **Biber braucht einen zusätzlichen Schritt**, weil das separat gemountete TeX-Live-Verzeichnis die im Image vorverdrahteten `PATH`-Einträge überschreibt.

Datei `scripts/050_setup_biber.sh` anlegen:

```
#!/bin/bash
ln -sf /usr/local/texlive/<JAHR>/bin/x86_64-linux/biber /usr/local/bin/biber
```

`<JAHR>` an die tatsächlich installierte TeX-Live-Version anpassen (z. B. `2026`). Dieses Skript wird über `/etc/my_init.d/050_setup_biber.sh` beim Container-Start ausgeführt (Hook-Mechanismus von `phusion baseimage`, auf dem auch das offizielle Image basiert).

---

5. Phase 3 – `docker-compose.yml`

Verzeichnisstruktur auf dem Server, z. B. unter `/opt/overleaf/`:

```
/opt/overleaf/
├── docker-compose.yml
├── overleaf_data/     (wird beim ersten Start automatisch angelegt)
```

```
├── mongo_data/
├── redis_data/
└── scripts/
    └── 050_setup_biber.sh
```

`docker-compose.yml`:

```
services:
  sharelatex:
    image: sharelatex/sharelatex
    container_name: sharelatex
    restart: always
    depends_on:
      - mongo
      - redis
    ports:
      - "80:80"
    links:
      - mongo
      - redis
    volumes:
      - ./overleaf_data:/var/lib/overleaf
      - /usr/local/texlive/2026:/usr/local/texlive/2026   # Pfad an installierte Version
anpassen
      - ./scripts/050_setup_biber.sh:/etc/my_init.d/050_setup_biber.sh
    environment:
      OVERLEAF_APP_NAME: "<Name der Institution>"
      OVERLEAF_MONGO_URL: mongodb://mongo:27017/sharelatex?replicaSet=rs0
      OVERLEAF_REDIS_HOST: redis
      REDIS_HOST: redis


  mongo:
    image: mongo:8.0
    container_name: mongo
    restart: always
    # WICHTIG: Overleaf braucht zwingend ein Replica-Set, auch bei einem einzelnen Mongo-
Knoten
    command: "--replSet rs0 --bind_ip_all"
    volumes:
      - ./mongo_data:/data/db
```

```
  redis:
    image: redis:6.2
    container_name: redis
    restart: always
    volumes:
      - ./redis_data:/data
```

Kein `EXTERNAL_AUTH`/OIDC-Block nötig, da lokale Accounts ausreichen. TLS/Reverse-Proxy ist hier bewusst nicht enthalten — dafür separat einen Reverse Proxy (z. B. nginx) mit intern gültigem TLS-Zertifikat vor Port 80 setzen, auch im abgeschotteten Netz.

---

6. Phase 4 – Start, Mongo-Replica-Set, Admin-Account

```
cd /opt/overleaf
docker compose up -d
```

Nach dem ersten Start des Mongo-Containers muss das Replica-Set einmalig initialisiert werden:

```
docker exec -it mongo mongosh --eval "rs.initiate({_id: 'rs0', members: [{_id: 0, host: 'mongo:27017'}]})"
```

Ersten Admin-Account anlegen:

```
docker exec sharelatex /bin/bash -ce \
  "cd /overleaf/services/web && node modules/server-ce-scripts/scripts/create-user --admin --email=admin@behoerde.de"
```

Die Ausgabe enthält einen Link, über den das Passwort für diesen Account gesetzt wird. Weitere Nutzer-Accounts (≤10 geplant): Admin legt sie im Admin-Panel an, jeder Nutzer erhält einen Einladungslink per Mail zum Setzen des eigenen Passworts.

---

7. Phase 5 – Funktionstests vor der Netztrennung

Vor dem endgültigen Abschotten unbedingt testen:

1. **Einfacher Compile-Test** (Standard-LaTeX-Dokument ohne Bibliografie).
2. **BibTeX-Test**: Testprojekt mit `\bibliographystyle{...}` + klassischem `.bib`, erfolgreicher Compile-Lauf im Log prüfen.
3. **Biber-Test**: Testprojekt mit `\usepackage[backend=biber]{biblatex}`, im Compile-Log prüfen, dass `biber` tatsächlich aufgerufen wird (nicht "command not found").
4. **Offline-Verhalten**: Internetzugang des Servers testweise kappen und die Web-UI erneut durchklicken — prüfen, ob irgendwo externe Ressourcen (Fonts, Assets, Update-Checks) fehlschlagen. Erst danach endgültig abschotten.

---

8. Phase 6 – Endgültige Netztrennung

1. Proxy-Freigaben (Abschnitt 3.3) beim Netzwerk-/Security-Team zurückziehen lassen.
2. Docker-Proxy-Konfiguration wieder entfernen:

   ```
   sudo rm /etc/systemd/system/docker.service.d/http-proxy.conf
   sudo systemctl daemon-reload
   sudo systemctl restart docker
   ```

3. Server aus dem Netzwerksegment mit Internetzugang herausnehmen / entsprechende Firewall-Regeln final auf "kein Outbound" setzen.

---

9. Branding / Vorlage

- `OVERLEAF_APP_NAME` deckt den Produktnamen ab.
- Logo/CSS-Anpassung erfordert Überschreiben von Assets im Image bzw. Mounten eigener Dateien an bestimmten Pfaden — **vor Produktivsetzung an einem Testsystem verifizieren**, nicht ungeprüft übernehmen.

- Eine institutsweite Standard-Vorlage für neue Projekte lässt sich nicht per Env-Var setzen, sondern nur über den Admin-Bereich bzw. eine vorbereitete Projekt-Kopie — ebenfalls vorab testen.

---

10. Backup

Kein automatisches Backup-Konzept im Referenzaufbau vorhanden — für den Produktivbetrieb mindestens einplanen:

- Regelmäßiger `mongodump` der Projekt-/Nutzerdatenbank.
- Snapshot/Kopie der Daten-Volumes (`overleaf_data`, `mongo_data`, `redis_data`).
- TeX-Live-Volume ändert sich nach der Installation nicht mehr und muss nicht laufend gesichert werden, nur einmalig dokumentiert/archiviert werden (falls der Server neu aufgesetzt werden muss, ohne erneut online zu gehen).

---

11. Offene Punkte für die Entscheidung / Dokumentation gegenüber Vorgesetzten

Thema	Entscheidung	Begründung
Fork vs. offizielles Image	offizielles `sharelatex/sharelatex`	Herstellersupport, kleinere Angriffsfläche, keine SSO-Anforderung mehr, die den Fork nötig gemacht hätte
Toolkit vs. manuelles `docker-compose.yml`	manuelles Compose	direkter nachvollziehbar, weniger bewegliche Teile, keine zusätzliche GitHub-Freigabe nötig
Server Pro vs. Community Edition	Community Edition	kein Budget vorhanden
Sandboxed Compiles	nicht vorhanden, akzeptiertes Restrisiko	kleine, bekannte Nutzergruppe (Studierende, eigene Abschlussarbeiten)
Nutzerverwaltung	lokal, manuell, ≤10 Accounts	kein AD-Anschluss zum Start nötig/gewünscht
BibTeX/Biber	scheme-full TeX Live + Biber-Symlink-Skript	vermeidet Nachinstallation im abgeschotteten Betrieb

Thema	Entscheidung	Begründung
Container-Engine	Docker (kein unterstützter Weg ohne Container)	Overleaf CE ist architektonisch auf Microservices im Container-Image angewiesen; nativ nicht von Overleaf unterstützt

Quellen

- Overleaf on-premises Doku – Server Pro vs. Community Edition
- Overleaf – Creating and managing users
- Overleaf GitHub Issue – kein unterstützter Non-Docker-Weg
- CTAN Mirror-Liste

Revision #2
Created 2026-07-21 13:23:36 UTC by Denode
Updated 2026-07-21 13:25:44 UTC by Denode