ADMIN:DOCS

Ubuntu 24.04: Stirling PDF installieren und produktionsreif betreiben

Leitfaden zur Installation von Stirling PDF auf Ubuntu 24.04 LTS mit Docker Compose: OCR-Einrichtung, Reverse Proxy mit TLS, Absicherung und Betriebsüberwachung.

Kategorie: Serverumgebungen **Veröffentlicht:** 09.10.2024 **Zuletzt geprüft:** 06.09.2026

PDF-Dokumente gehören in Unternehmen und Behörden zu den sensibelsten Datenbeständen: Arbeitsverträge, Rechnungen, Steuerunterlagen und interne Berichte dürfen aus Gründen des Datenschutzes und der DSGVO niemals unkontrolliert an externe Cloud-Dienste übertragen werden. Webbasierte Tools wie Smallpdf oder Adobe Cloud lösen zwar alltägliche Aufgaben wie das Zusammenführen oder Schwärzen von Seiten, verletzen im geschäftlichen Umfeld jedoch regelmäßig Compliance-Richtlinien.

Stirling PDF löst dieses Dilemma als vollständig quelloffene, lokal gehostete Komplettlösung. Die Anwendung läuft als leichtgewichtiger Docker-Container, verarbeitet alle Dokumente lokal im Arbeitsspeicher des eigenen Servers und hinterlässt nach Abschluss der Bearbeitung keine persistenten Dateireste auf externen Plattformen.

Die Bereitstellung von Stirling PDF erfolgt auf einem Server mit **Ubuntu 24.04 LTS (Noble Numbat)**: mit integrierter OCR-Engine für deutsche und englische Texte, gehärtetem Nginx Reverse Proxy samt TLS-Terminierung sowie reproduzierbaren Update- und Backup-Routinen für den Produktivbetrieb.

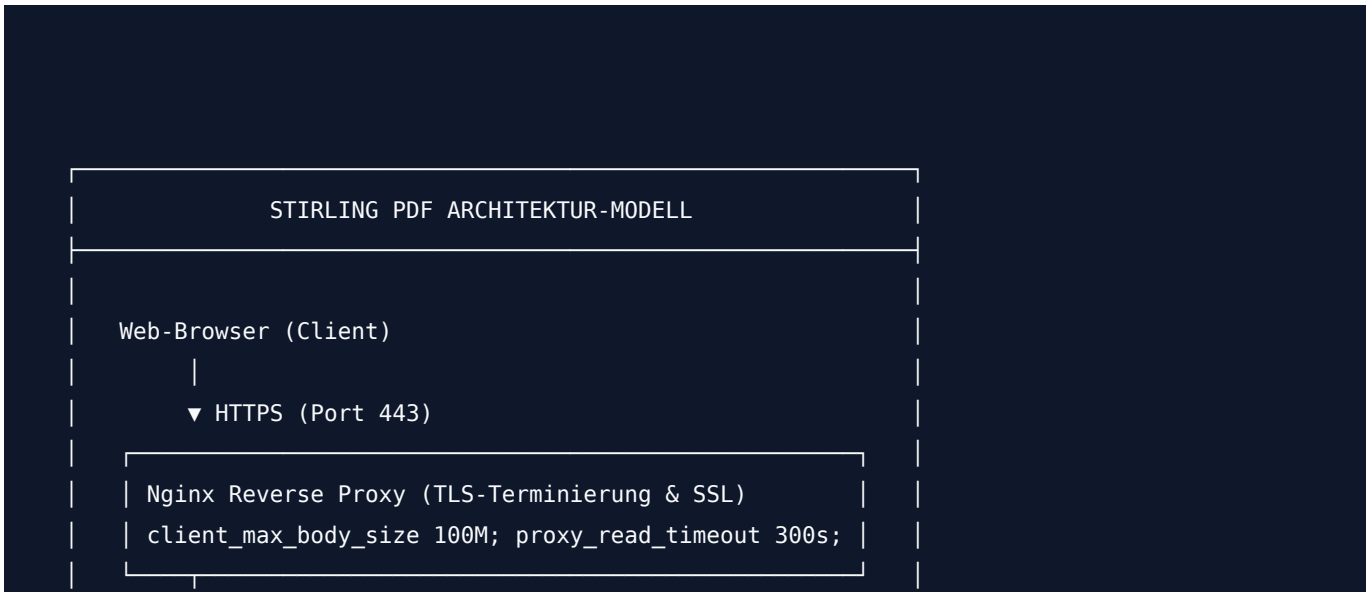
Quelle: https://admindocs.de/de/serverumgebungen/ubuntu-24-04-stirling-pdf-installieren

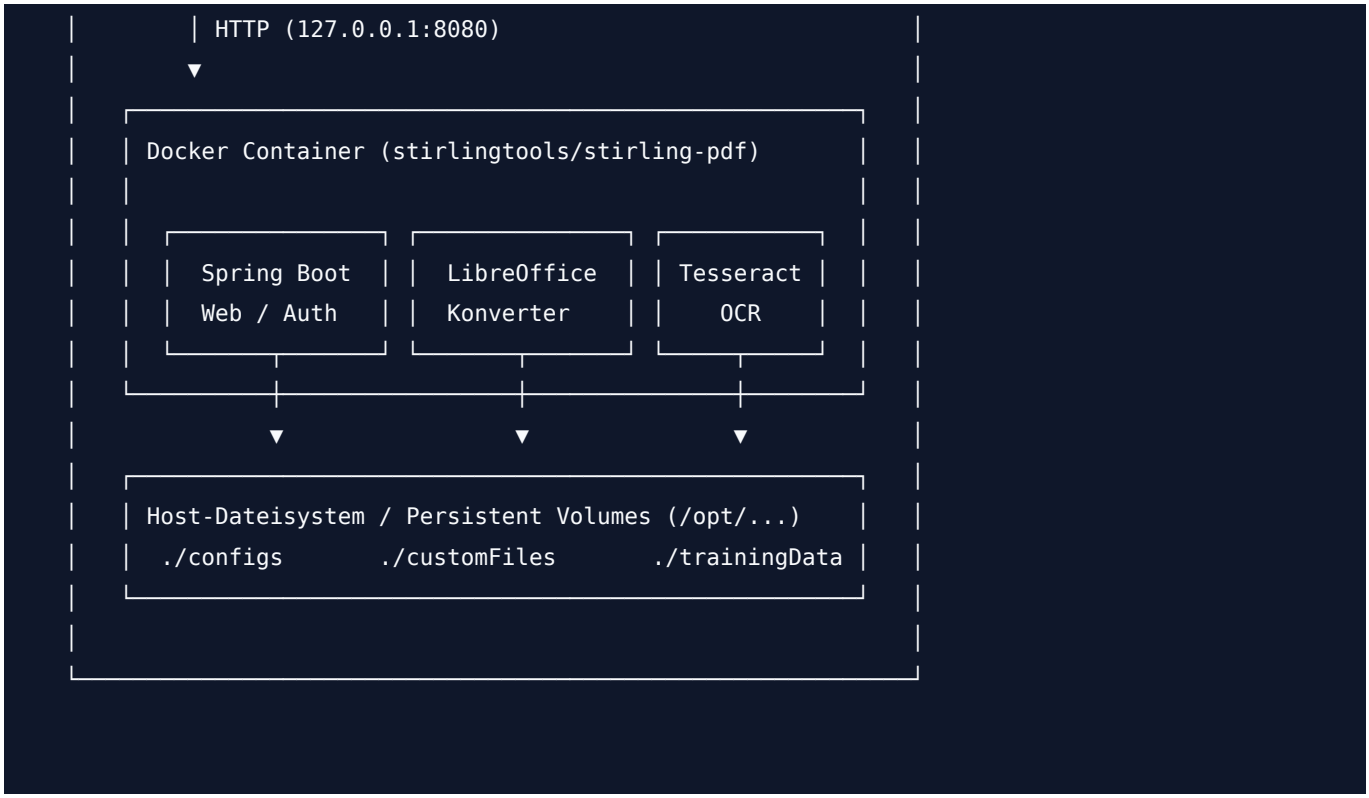
> **Hinweis:** Eine funktionierende Online-Demo der Software stellt das Projekt unter pdf.ipx64.xyz bereit. Für die eigene Infrastruktur nutzen wir die offizielle Container-Distribution.

Architektur und Komponenten

Stirling PDF basiert im Kern auf einer Java-Anwendung (Spring Boot) und orchestriert unter der Haube spezialisierte Linux-Kommandozeilenwerkzeuge. Anstatt das Rad neu zu erfinden, kapselt der Container bewährte Open-Source-Bibliotheken:

- **Apache PDFBox & OpenPDF:** Übernehmen Low-Level-PDF-Operationen (Seiten trennen, verbinden, drehen, Metadaten editieren).
- **LibreOffice (Headless):** Konvertiert Office-Formate (DOCX, ODT, PPTX, XLSX) verlustfrei in standardkonforme PDF-Dateien.
- **Tesseract OCR & OCRmyPDF:** Durchsuchen gescannte Dokumente, erkennen Textschichten und binden durchsuchbare OCR-Texte in bestehende PDFs ein.
- **Ghostscript & QPDF:** Optimieren Dateigrößen, reparieren beschädigte PDF-Strukturen und entschlüsseln kennwortgeschützte Dateien.





Die Trennung zwischen Reverse Proxy auf dem Host und dem Stirling-Container auf `127.0.0.1` stellt sicher, dass unverschlüsselter HTTP-Traffic niemals ungeschützt ins externe Netzwerk gerät.

Systemvoraussetzungen und Ressourcenplanung

Bevor wir mit der Bereitstellung beginnen, prüfen wir die Hardware-Ressourcen des Zielservers. PDF-Operationen mit LibreOffice und optischer Zeichenerkennung (OCR) sind CPU- und arbeitsspeicherintensiv.

Minimale vs. Empfohlene Systemressourcen:

Ressource	Minimaler Betrieb (1–2 Nutzer)	Empfohlen für Produktivbetrieb & OCR
CPU	2 vCPUs	4 vCPUs (beschleunigt parallele OCR-Jobs)
Arbeitsspeicher	2 GB RAM	4 bis 8 GB RAM (Tesseract + JVM)
Festplatte	10 GB freier Speicherplatz	25 GB SSD/NVMe (für Temp-Dateien & Docker-Images)

Ressource	Minimaler Betrieb (1–2 Nutzer)	Empfohlen für Produktivbetrieb & OCR
Betriebssystem	Ubuntu 24.04 LTS (x86_64 oder ARM64)	Ubuntu 24.04 LTS (x86_64 oder ARM64)

> ⚠ **Achtung vor dem OOM-Killer:** Wenn Stirling PDF ein 200-seitiges PDF per OCR verarbeitet oder komplexe Word-Dokumente über LibreOffice rendert, kann der Speicherverbrauch kurzzeitig um 1,5 bis 2 GB ansteigen. Auf Systemen mit lediglich 2 GB RAM muss zwingend eine aktive Swap-Datei vorhanden sein, da der Linux-Kernel den Container andernfalls per Out-of-Memory-Signal (`SIGKILL`) beendet.

Schritt 1: System aktualisieren und Docker installieren

Ubuntu 24.04 liefert in seinen Standard-Paketquellen noch immer das alte Paket `docker.io` sowie die veraltete Python-Version von Docker Compose aus. Für einen stabilen und sicheren Betrieb verwenden wir das offizielle Repository von Docker Inc. mit der modernen Compose-v2-Erweiterung (`docker-compose-plugin`).

Paketquellen aktualisieren und Basistools installieren:

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y ca-certificates curl gnupg lsb-release
```

Offiziellen Docker-GPG-Schlüssel hinzufügen und Repository konfigurieren:

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Docker Engine und Docker Compose v2 installieren:

```
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin
```

Dienststatus und Autostart validieren:

```
sudo systemctl enable --now docker
docker compose version
```

```
# Erwartete Ausgabe:
Docker Compose version v2.29.x (oder neuer)
```

Schritt 2: Verzeichnisstruktur und Berechtigungen anlegen

Wir legen die Container-Konfiguration strukturiert unter `/opt/stirling-pdf` ab. Alle persistenten Daten, Sprachmodelle und benutzerdefinierten Einstellungen verbleiben isoliert in dedizierten Unterverzeichnissen:

```
sudo mkdir -p /opt/stirling-pdf/{configs,customFiles,trainingData,pipeline}
cd /opt/stirling-pdf
```

Bedeutung der Verzeichnisse:

- `configs/`: Beherbergt die Anwendungskonfiguration (`settings.yml`), Datenbanken für Benutzeraccounts und API-Schlüssel.
- `customFiles/`: Ermöglicht das Hinterlegen eigener Logos, CSS-Anpassungen oder Wasserzeichen-Vorlagen.
- `trainingData/`: Speicherort für Tesseract-OCR-Sprachpakete (`.traineddata`), z. B. Deutsch (`deu`) und Englisch (`eng`).
- `pipeline/`: Optionaler Ordner für automatisierte Verarbeitungs-Pipelines (z. B. automatisches OCR für abgelegte Scans).

Schritt 3: OCR-Sprachdateien herunterladen

Standardmäßig liefert das Basis-Image von Stirling PDF englische Sprachdaten mit. Um Scans auf Deutsch fehlerfrei verarbeiten zu können (inklusive Umlauten `ä`, `ö`, `ü` und `ß`), laden wir das optimierte Fast-Tessdata-Modell direkt in unser Volume:

```
cd /opt/stirling-pdf/trainingData
sudo curl -fsSL -O https://github.com/tesseract-ocr/tessdata_fast/raw/main/deu.traineddata
sudo curl -fsSL -O https://github.com/tesseract-ocr/tessdata_fast/raw/main/eng.traineddata
cd /opt/stirling-pdf
```

> **Tipp zur Modellqualität:** Für Archivierungsanforderungen mit maximaler Zeichenpräzision kann alternativ das Repository `tessdata_best` verwendet werden. Das `tessdata_fast`-Modell benötigt jedoch nur rund ein Viertel des Arbeitsspeichers und liefert bei Standardschriftarten nahezu identische Erkennungsraten bei signifikant höherer Durchsatzgeschwindigkeit.

Schritt 4: Produktionsreife Docker Compose Konfiguration

Stirling PDF bietet zwei primäre Image-Varianten an:

1. `stirlingtools/stirling-pdf:latest`: Standard-Image mit vollständigem Funktionsumfang (LibreOffice, OCR, PDFBox).
2. `stirlingtools/stirling-pdf:latest-fat`: Enthält bereits sämtliche weltweiten Tesseract-Sprachdateien (erfordert über 4 GB Speicherplatz beim Pull).

Wir nutzen das Standard-Image und binden gezielt die benötigten Sprachmodelle über das gemountete `trainingData`-Volume ein.

Erstelle die Datei `/opt/stirling-pdf/docker-compose.yml`:

```
sudo nano /opt/stirling-pdf/docker-compose.yml
```

Füge folgende Konfiguration ein:

```
services:
  stirling-pdf:
    image: stirlingtools/stirling-pdf:latest
    container_name: stirling-pdf
    restart: unless-stopped
    ports:
      # Nur auf Localhost binden – Absicherung über Nginx Reverse Proxy
      - "127.0.0.1:8080:8080"
    volumes:
      - ./trainingData:/usr/share/tessdata:rw
      - ./configs:/configs:rw
      - ./customFiles:/customFiles:rw
      - ./pipeline:/pipeline:rw
    environment:
      # Benutzeroberfläche und Lokalisierung
      - SYSTEM_DEFAULTLOCALE=de-DE
      - UI_APPNAME=AdminDocs PDF Studio
      # Sicherheits- und Authentifizierungsoptionen
      - DOCKER_ENABLE_SECURITY=true
      - SECURITY_ENABLE_LOGIN=true
      - SECURITY_CSRF_DISABLED=false
      # Performance- und Ressourceneinstellungen
      - INSTALL_BOOK_AND_ADVANCED_HTML_OPS=false
      - SYSTEM_MAXFILESIZE=100
    deploy:
      resources:
        limits:
          cpus: '3.00'
          memory: 3500M
        reservations:
```

```
            memory: 1024M
      healthcheck:
        test: ["CMD-SHELL", "curl -f http://localhost:8080/api/v1/info/status || exit 1"]
        interval: 30s
        timeout: 10s
        retries: 3
        start_period: 40s
```

Erklärung der zentralen Parameter:

- `127.0.0.1:8080:8080`: Bindet den Service exklusiv an den lokalen Loopback-Adapter. Externe Anfragen über den Port 8080 werden direkt verworfen und müssen zwingend den TLS-gesicherten Proxy passieren.
- `DOCKER_ENABLE_SECURITY=true` & `SECURITY_ENABLE_LOGIN=true`: Aktiviert das integrierte Rollen- und Benutzersystem von Stirling PDF.
- `SYSTEM_MAXFILESIZE=100`: Definiert das Upload-Limit für Eingabedateien auf 100 Megabyte.
- `limits.memory: 3500M`: Schützt den Host vor unkontrolliertem Speicherhunger bei fehlerhaften oder manipulierten PDF-Dateien.

Schritt 5: Container starten und initialen Administrator erzeugen

Wir starten den Container im Hintergrund:

```
cd /opt/stirling-pdf
sudo docker compose up -d
```

Startvorgang in den Logs verfolgen:

```
sudo docker compose logs -f stirling-pdf
```

Beim ersten Start mit aktivierter Sicherheitsoption (`DOCKER_ENABLE_SECURITY=true`) erzeugt Stirling PDF automatisch einen temporären Administrator-Account. Halte in den Logs nach folgender Zeichenfolge Ausschau:

```
#################################################################
# Generated Admin Username: admin                               #
# Generated Admin Password: <generiertes-einmal-passwort>       #
#################################################################
```

Kopiere dieses Initialpasswort in die Zwischenablage. Wir nutzen es im Webinterface, um sofort ein permanentes Kennwort zu vergeben.

> **Hinweis zum Initialpasswort:** Falls das Passwort in den Logs übersehen wurde, kann es jederzeit in der Datei `/opt/stirling-pdf/configs/settings.yml` eingesehen oder durch Stoppen des Containers und Zurücksetzen der Konfiguration neu generiert werden.

Schritt 6: Absicherung mit Nginx Reverse Proxy und Let's Encrypt

Für den produktiven Einsatz im Intranet oder Internet binden wir Stirling PDF über einen Nginx Reverse Proxy mit sicherem TLS-Zertifikat an.

Nginx und Certbot installieren:

```
sudo apt install -y nginx certbot python3-certbot-nginx
```

Nginx-Konfiguration für die Domain anlegen:

Ersetze `pdf.example.com` durch deinen gewünschten Domainnamen:

```
sudo nano /etc/nginx/sites-available/stirling-pdf.conf
```

Füge folgenden Server-Block ein:

```
server {
    listen 80;
    listen [::]:80;
    server_name pdf.example.com;

    # Großzügige Limits für große Dokumenten-Uploads
    client_max_body_size 100M;
    client_body_timeout 300s;

    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_http_version 1.1;

        # Header zur Weitergabe der Client-Informationen
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
```

```
        # WebSocket-Unterstützung für Fortschrittsanzeigen
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";

        # Timeouts für rechenintensive OCR- und Konvertierungs-Jobs
        proxy_connect_timeout 90s;
        proxy_send_timeout 300s;
        proxy_read_timeout 300s;
    }
}
```

Warum client_max_body_size und Timeouts unverzichtbar sind:

Standardmäßig beschränkt Nginx Anfragen auf 1 Megabyte (`client_max_body_size 1M`). Jeder Upload eines gescannten PDFs oder eines umfangreichen Handbuchs würde andernfalls mit dem Fehler `413 Request Entity Too Large` abgewiesen werden. Die verlängerten Timeouts (`proxy_read_timeout 300s`) verhindern Verbindungsabbrüche (`504 Gateway Timeout`), wenn Tesseract Hunderte von Seiten analysiert.

Site aktivieren und Konfiguration prüfen:

```
sudo ln -s /etc/nginx/sites-available/stirling-pdf.conf /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

TLS-Zertifikat mit Let's Encrypt erzeugen:

```
sudo certbot --nginx -d pdf.example.com
```

Certbot erweitert die Nginx-Konfiguration automatisch um TLSv1.2/TLSv1.3-Direktiven und richtet eine automatische Zertifikatsverlängerung per systemd-Timer ein.

Schritt 7: Firewall mit UFW konfigurieren

Wir stellen sicher, dass ausschließlich die sicheren Webports nach außen erreichbar sind:

```
sudo ufw default deny incoming
sudo ufw default allow outgoing
sudo ufw allow ssh
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp
sudo ufw enable
```

```
# Status kontrollieren:
sudo ufw status verbose
```

Der interne Port `8080` ist dank unserer Compose-Bindung auf `127.0.0.1` isoliert und taucht in den externen Firewall-Regeln gar nicht erst auf.

Schritt 8: Verifikation und Systemüberprüfung

Nach dem Setup überprüfen wir den Betriebszustand aller Komponenten über die Linux-Konsole.

1. Port-Bindung kontrollieren:

```
ss -tulpn | grep 8080
```

```
# Erwartetes Resultat (muss 127.0.0.1 zeigen, keinesfalls 0.0.0.0):
tcp   LISTEN 0      4096   127.0.0.1:8080       0.0.0.0:*    users:(("docker-
proxy",pid=...))
```

2. Docker Healthcheck prüfen:

```
docker compose -f /opt/stirling-pdf/docker-compose.yml ps
```

```
# Erwartetes Ergebnis (Status "healthy"):
NAME           IMAGE                              COMMAND                  SERVICE
STATUS
stirling-pdf   stirlingtools/stirling-pdf:latest "tini -- ./entrypoin…"   stirling-pdf
Up 5 minutes (healthy)
```

3. API-Status und Tesseract-Unterstützung abfragen:

```
curl -s http://127.0.0.1:8080/api/v1/info/status
```

Die JSON-Antwort muss den Status `UP` melden und die installierten OCR-Sprachen auflisten.

Wartung, Updates und Backup-Strategie

Ein professioneller Betrieb erfordert verlässliche Wartungsroutinen für Container-Updates und Konfigurations-Backups.

Container-Updates durchführen

Da Stirling PDF aktiv weiterentwickelt wird, sollten monatliche Updates eingeplant werden:

```
cd /opt/stirling-pdf
sudo docker compose pull
sudo docker compose down
sudo docker compose up -d
sudo docker image prune -f
```

Konsistentes Backup der Konfiguration erstellen

Da die eigentlichen PDF-Dateien nach der Bearbeitung sofort aus dem temporären Speicher gelöscht werden, müssen wir für das Disaster-Recovery lediglich die Konfigurationsdaten und Benutzereinstellungen sichern:

```
#!/bin/bash
# Backup-Skript für Stirling PDF
BACKUP_DIR="/var/backups/stirling-pdf"
DATE=$(date +%Y%m%d_%H%M%S)

mkdir -p "$BACKUP_DIR"
tar -czf "$BACKUP_DIR/stirling_config_$DATE.tar.gz" -C /opt/stirling-pdf configs
customFiles trainingData

# Backups älter als 30 Tage bereinigen
find "$BACKUP_DIR" -type f -name "*.tar.gz" -mtime +30 -delete
```

Dieses Skript kann als täglicher Cronjob unter `/etc/cron.daily/backup-stirling-pdf` hinterlegt werden.

Typische Fehler und Troubleshooting

In der Praxis können bei der PDF-Verarbeitung spezifische Fehlerzustände auftreten:

1. Fehler 413: Request Entity Too Large

- **Symptom:** Beim Hochladen eines PDFs mit mehr als 1 MB bricht der Browser mit HTTP 413 ab.
- **Ursache:** Die Nginx-Direktive `client_max_body_size` fehlt oder ist zu niedrig angesetzt.
- **Lösung:** In `/etc/nginx/sites-available/stirling-pdf.conf` den Wert `client_max_body_size 100M;` eintragen und `sudo systemctl reload nginx` ausführen.

2. Tesseract erkennt keine deutschen Umlaute

- **Symptom:** OCR-Texte enthalten Hieroglyphen oder Leerzeichen anstelle von `ä`, `ö`, `ü`.
- **Ursache:** Die Sprachdatei `deu.traineddata` wurde nicht im gemounteten Verzeichnis gefunden oder hat falsche Dateiberechtigungen.

- **Diagnose und Behebung:**

```
ls -lh /opt/stirling-pdf/trainingData/
sudo chmod 644 /opt/stirling-pdf/trainingData/*.traineddata
sudo docker compose restart stirling-pdf
```

3. Container wird während OCR-Verarbeitung unerwartet beendet

- **Symptom:** `docker compose ps` zeigt `Exited (137)`.
- **Ursache:** Exit-Code 137 bedeutet `SIGKILL` durch den Linux Out-of-Memory (OOM) Killer.
- **Diagnose:** `dmesg -T | grep -i oom` prüfen.
- **Lösung:** Erhöhe das Speicherlimit in `docker-compose.yml` (`memory: 4096M`) oder richte eine 4 GB Swap-Datei auf dem Host ein:

```
sudo fallocate -l 4G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

Befehlsreferenz (Cheatsheet)

Die wichtigsten operativen Befehle für Verwaltung und Fehlerdiagnose auf einen Blick:

Aufgabe / Befehl	Funktion / Erläuterung

`docker compose -f /opt/stirling-pdf/docker-compose.yml up -d`	Startet den Stirling PDF Stack im Hintergrund
`docker compose -f /opt/stirling-pdf/docker-compose.yml logs -f`	Zeigt Echtzeit-Logs der Anwendung und des Webservers
`docker compose -f /opt/stirling-pdf/docker-compose.yml restart`	Startet den Container neu (z. B. nach Konfigurationsänderung)
`docker stats stirling-pdf`	Überwacht Live-CPU- und Speicherauslastung des Containers
`nginx -t && systemctl reload nginx`	Validiert Nginx-Reverse-Proxy-Konfiguration und lädt Regeln neu
`certbot renew --dry-run`	Simuliert die automatische Verlängerung des TLS-Zertifikats
`ss -tulpn	grep 8080`	Überprüft, ob Port 8080 ausschließlich auf `127.0.0.1` lauscht

Weiterführende Ressourcen

Die folgenden Dokumentationen bieten vertiefende Informationen zu den eingesetzten Technologien und Komponenten:

Ressource / Dokumentation	Beschreibung
Offizielle Stirling-PDF-Dokumentation	Ausführliche Dokumentation zu allen Parametern, API-Endpunkten und Pipeline-Funktionen
Stirling-Tools GitHub Repository	Offizieller Quellcode, Issue-Tracker und Release-Notes
Docker Engine Dokumentation für Ubuntu	Best Practices zur Installation und Pflege von Docker unter Ubuntu Linux
Tesseract OCR Projekt	Hintergrundinformationen zur Texterkennungs-Engine und Sprachmodellen

Ressource / Dokumentation	Beschreibung
AdminDocs: Docker und Docker Compose Leitfaden	Grundlegende Container-Orchestrierung und Absicherung unter Linux
Portainer CE Web-GUI	Grafische Verwaltung von Docker-Containern und Stacks unter Ubuntu 24.04 LTS

Fazit

Mit dem hier aufgebauten Setup steht auf Ubuntu 24.04 LTS eine vollwertige, DSGVO-konforme Dokumentenzentrale bereit. Durch die strikte Kapselung im Docker-Container, die Entkopplung über einen gehärteten Nginx Reverse Proxy und die Begrenzung auf den lokalen Loopback-Adapter bleibt die Sicherheitsarchitektur transparent und wartungsarm.

> **Praxistipp für Administratoren:** Richte in Stirling PDF unter den Admin-Einstellungen Benutzer-Quotas und Session-Timeouts ein. Werden vertrauliche Dokumente von Mitarbeitern im Browser bearbeitet, sollten die temporären Browser-Caches nach Sitzungsende automatisch invalidiert werden. Aktiviere zudem die Option zur automatischen Metadaten-Bereinigung, um versteckte Autoren- und Drucker-Informationen vor der Weitergabe von PDF-Dokumenten verlässlich zu entfernen.

Damit entfällt die Notwendigkeit für ungesicherte Drittanbieter-Webdienste im Unternehmensnetzwerk vollständig, während die Datenhoheit zu 100 Prozent auf der eigenen Serverinfrastruktur verbleibt.