

Stirling PDF

Im Gegensatz zu anderen Online-PDF-Tools mit einem vergleichbaren Funktionsumfang hat Stirling PDF den Vorteil, dass die Daten nicht in Kontakt mit fremden Servern kommen: Die Daten verlassen nicht das eigene Netzwerk, sondern die Anwendung lässt sich lokal (bspw. innerhalb einer VM auf einem NAS) betreiben. In diesem Beispiel erfolgt die Installation via Docker-Container in einer virtuellen Maschine, auf der sich zuvor nur ein frisch installiertes Betriebssystem Debian Bookworm 12.5 befindet.

- [Funktionsübersicht](#)
- [Installation](#)
 - [Konfiguration des Docker-Containers](#)
 - [Installation von Docker und Docker-Compose](#)
 - [Docker-Container erstellen](#)
 - [Sprachpakete hinzufügen](#)
- [Bereitstellung von Stirling-PDF](#)
 - [Zugang zur Anwendung](#)
 - [Ausgangslage und Zielsetzung](#)
 - [Zugang per ReverseProxy](#)

Funktionsübersicht

- PDFs in mehrere Dateien mit bestimmten Seitenzahlen aufteilen oder alle Seiten als einzelne Dateien extrahieren
- mehrere PDFs zu einer einzigen Datei zusammenführen
- Konvertieren von PDFs in und aus Bildern
- PDF-Seiten in verschiedenen Reihenfolgen umsortieren
- Hinzufügen von Bildern zu PDFs an bestimmten Stellen
- PDFs drehen (90-Grad-Schritte)
- PDFs komprimieren
- Hinzufügen und Entfernen von Passwörtern
- PDF-Berechtigungen festlegen
- Wasserzeichen hinzufügen
- Konvertieren aller gängigen Dateien in PDF
- Extraktion von Bildern aus PDFs
- OCR (Texterkennung)
- Bearbeitung von Meta-Daten
- Benutzerdefinierte Download-Optionen
- Parallele Dateiverarbeitung und Downloads

Installation

Diese Schritt-für-Schritt-Anleitung beschreibt alle Maßnahmen, die bei einem frisch installierten Debian Linux (hier: Version 12.5 Bookworm) nötig sind, um die Anwendung (extern) nutzen zu können.

Konfiguration des Docker-Containers

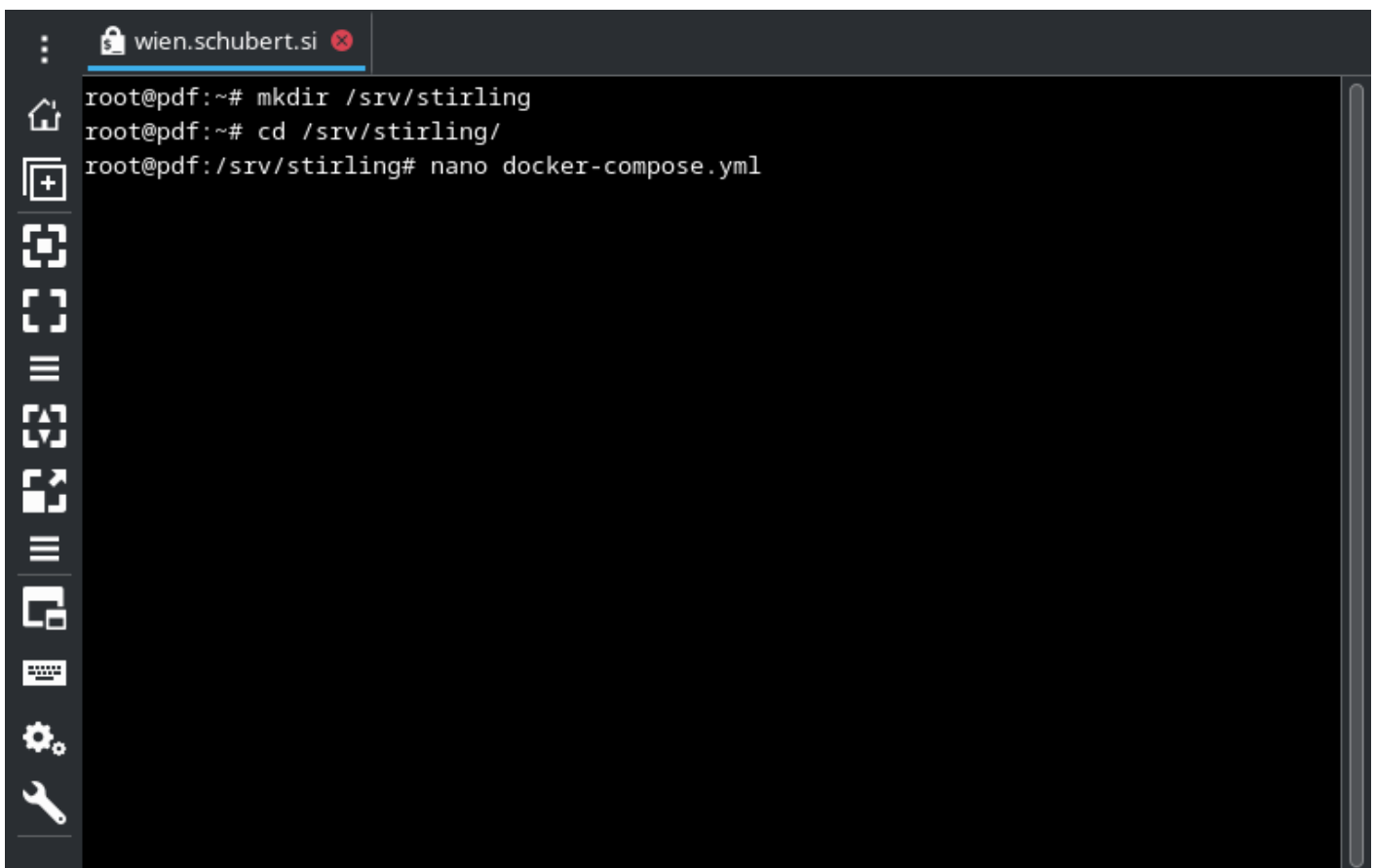
Wie bei [Docker-Compose](#) üblich wird der Container mit Hilfe einer speziellen Konfigurationsdatei (*docker-compose.yml*) spezifiziert.

Neben den Dateien des Docker-Containers sind zudem noch weitere Daten, bspw. in Form von Trainingsdaten für die Texterkennung, zu erwarten.

Um die Organisation der Dateien auf dem Server übersichtlich zu halten, empfiehlt es sich grundsätzlich für jeden (Server-)Dienst ein separates Verzeichnis zu nutzen.

In diesem Beispiel wird daher zunächst mit dem Befehl `mkdir /srv/stirling` ein neues Verzeichnis für Stirling-PDF erstellt und mittels `cd /srv/stirling/` dorthin gewechselt.

Dort wird dann mit Hilfe eines Text-Editors die Konfigurationsdatei angelegt. Bei der Verwendung von *nano* lautet der Befehl somit `nano docker-compose.yml`:



```
wien.schubert.si x
root@pdf:~# mkdir /srv/stirling
root@pdf:~# cd /srv/stirling/
root@pdf:/srv/stirling# nano docker-compose.yml
```

Die Konfigurationsdatei wird mit Daten nach diesem Schema erstellt:

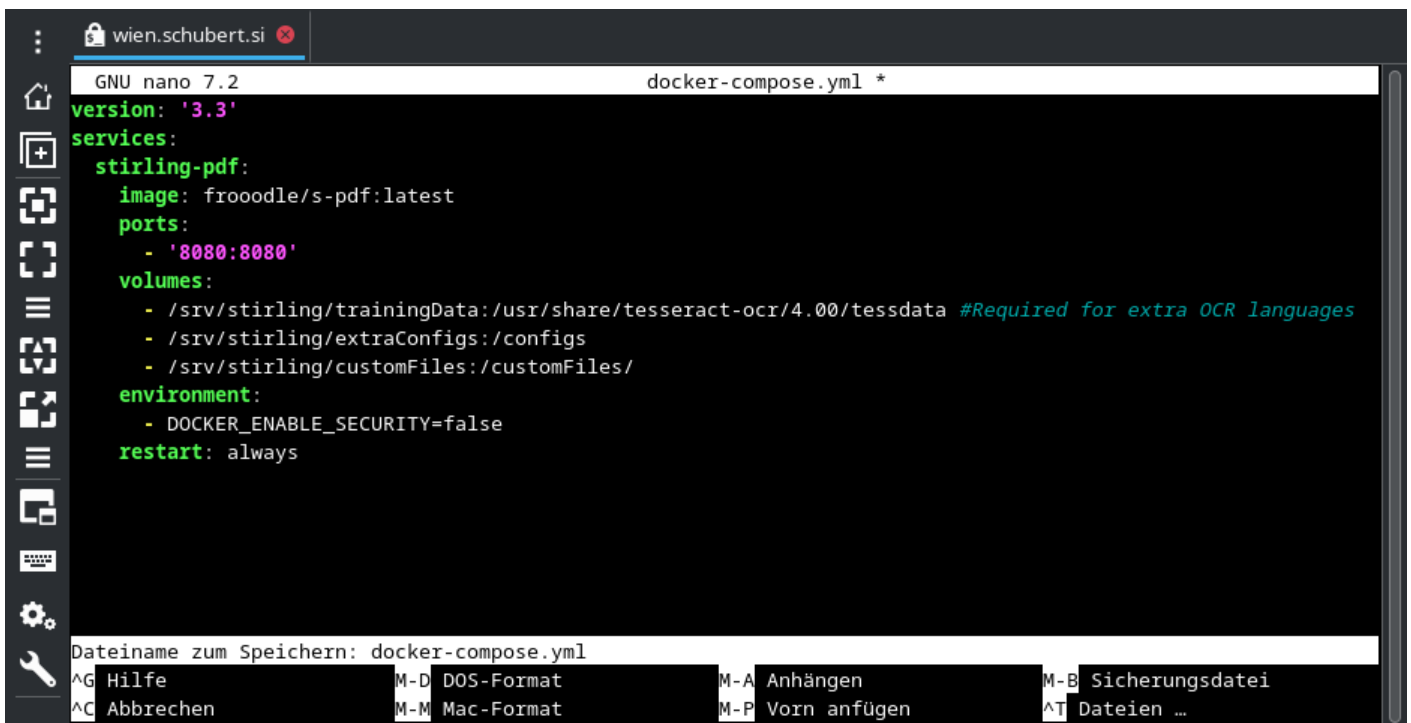
```
version: '3.3'
services:
  stirling-pdf:
    image: frooodle/s-pdf:latest
    ports:
      - '8080:8080'
    volumes:
      - /srv/stirling/trainingData:/usr/share/tesseract-ocr/4.00/tessdata #Required for extra OCR languages
      - /srv/stirling/extraConfigs:/configs
      - /srv/stirling/customFiles:/customFiles/
    environment:
      - DOCKER_ENABLE_SECURITY=false
    restart: always
```

Unter *ports* wird definiert, über welchen Port (links anzugeben) die Anwendung, die im Container unter dem rechts angegebenen Port läuft, später von außen zu erreichen ist. Der Eintrag `:8080` darf nicht verändert werden.

Soll Stirling-PDF später unter dem üblichen HTTP-Port 80 erreichbar sein, müsste der Eintrag auf `'80:8080'` geändert werden.

Bei *volumes* muss der Pfad angepasst werden, wenn nicht wie in diesem Beispiel `/srv/stirling` verwendet wird.

Mit der Tastenkombination `STRG` und `X` kann der Editor *nano* verlassen und die Datei nach Rückfrage gespeichert werden:



```
GNU nano 7.2 docker-compose.yml *
version: '3.3'
services:
  stirling-pdf:
    image: frooodle/s-pdf:latest
    ports:
      - '8080:8080'
    volumes:
      - /srv/stirling/trainingData:/usr/share/tesseract-ocr/4.00/tessdata #Required for extra OCR languages
      - /srv/stirling/extraConfigs:/configs
      - /srv/stirling/customFiles:/customFiles/
    environment:
      - DOCKER_ENABLE_SECURITY=false
    restart: always

Dateiname zum Speichern: docker-compose.yml
^G Hilfe M-D DOS-Format M-A Anhängen M-B Sicherungsdatei
^C Abbrechen M-M Mac-Format M-P Vorn anfügen ^T Dateien ...
```

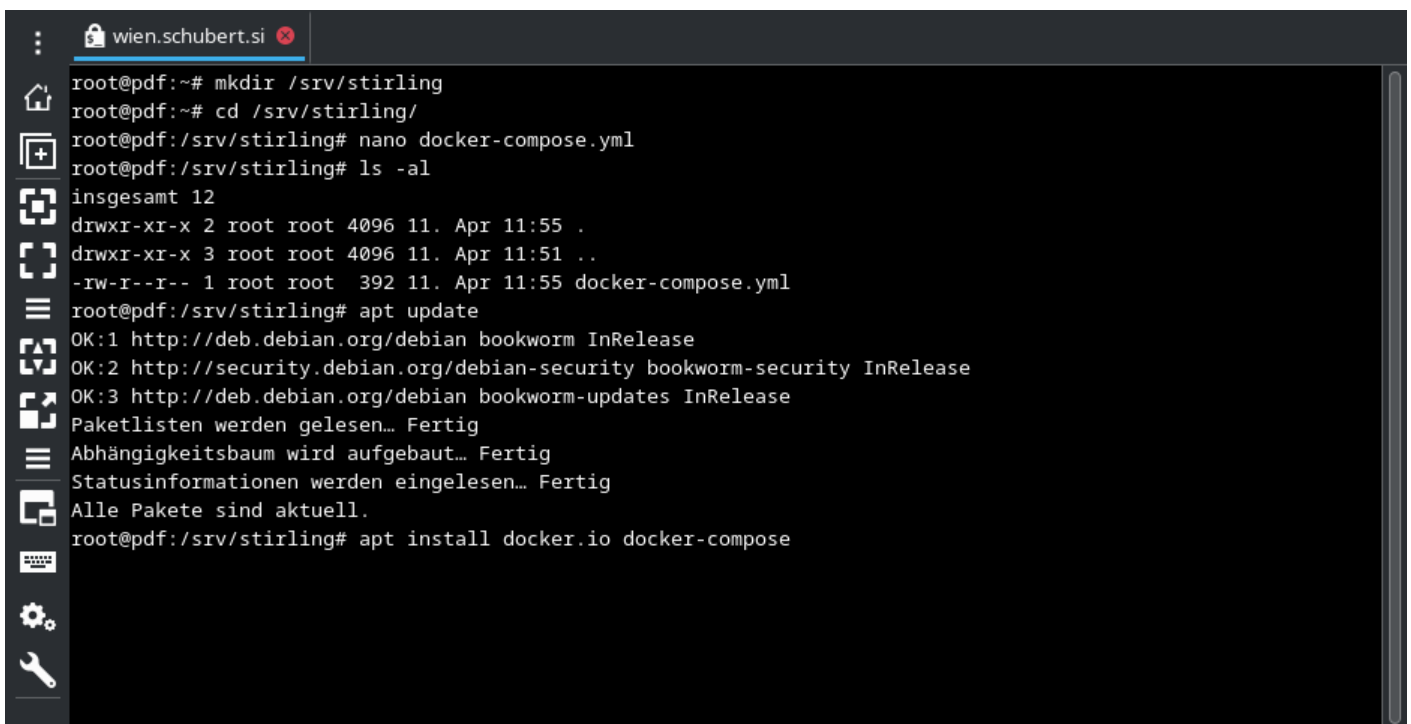
Die Definition des Docker-Containers ist damit abgeschlossen.

Installation von Docker und Docker-Compose

Auf einem frisch aufgesetzten Linux-Betriebssystem (wie hier Debian 12.5 Bookworm) ist in der Regel außer SSH kein weiterer Dienst - und somit auch kein [Docker](#) - verfügbar.

Daher gilt es zunächst Docker ("Betreiber des Containers") und Docker-Compose ("Zusammenbauer des Containers") in Betrieb zu nehmen.

Dafür werden als erstes die Paketquellen mittels `apt update` aktualisiert und ggf. Updates per `apt upgrade` (in diesem Beispiel nicht erforderlich, weil alle Pakete aktuell sind) eingespielt. Im Anschluss daran werden Docker und Docker-Compose durch den Befehl `apt install docker.io docker-compose` gemeinsam installiert:

A terminal window titled 'wien.schubert.si' showing a series of commands and their outputs. The user creates a directory, changes to it, creates a file, and lists its contents. Then, they run 'apt update' which shows three OK messages for repository updates. Finally, they run 'apt install docker.io docker-compose' which shows progress messages for reading package lists, building the dependency tree, and reading status information, concluding with 'Alle Pakete sind aktuell.'

```
root@pdf:~# mkdir /srv/stirling
root@pdf:~# cd /srv/stirling/
root@pdf:/srv/stirling# nano docker-compose.yml
root@pdf:/srv/stirling# ls -al
insgesamt 12
drwxr-xr-x 2 root root 4096 11. Apr 11:55 .
drwxr-xr-x 3 root root 4096 11. Apr 11:51 ..
-rw-r--r-- 1 root root 392 11. Apr 11:55 docker-compose.yml
root@pdf:/srv/stirling# apt update
OK:1 http://deb.debian.org/debian bookworm InRelease
OK:2 http://security.debian.org/debian-security bookworm-security InRelease
OK:3 http://deb.debian.org/debian bookworm-updates InRelease
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
Statusinformationen werden eingelesen... Fertig
Alle Pakete sind aktuell.
root@pdf:/srv/stirling# apt install docker.io docker-compose
```

Nach der Bestätigung mit der Taste `J` (oder `Y`, wenn keine dt. Lokalisation vorliegt) ...

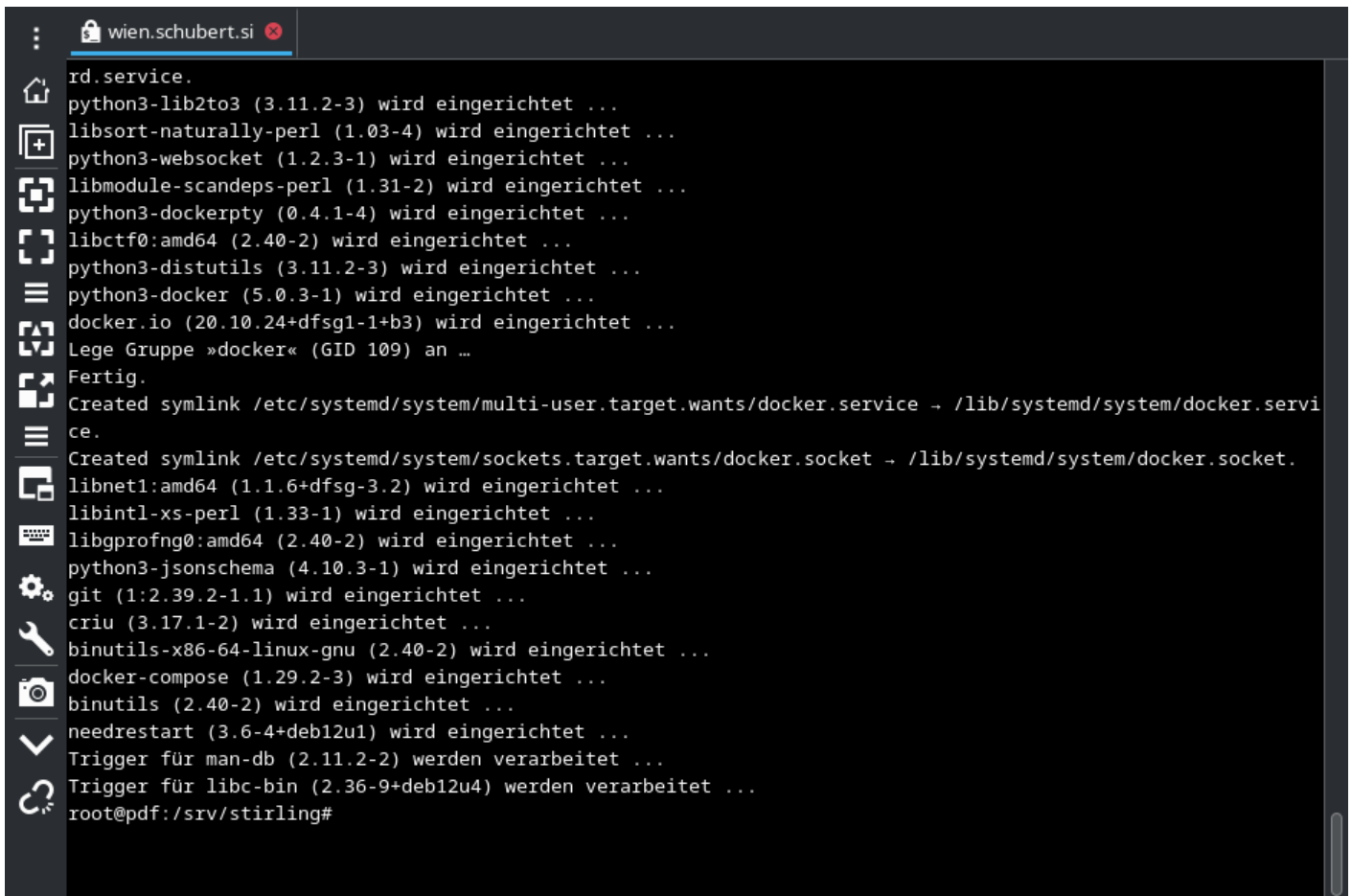
```
wien.schubert.si x
root@pdf:/srv/stirling# apt install docker.io docker-compose
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut... Fertig
Statusinformationen werden eingelesen... Fertig
Die folgenden zusätzlichen Pakete werden installiert:
 binutils binutils-common binutils-x86-64-linux-gnu cgroupfs-mount containerd criu git git-man libbinutils
 libc6-nobfd0 libc60 liberror-perl libgprofng0 libintl-perl libintl-xs-perl libmodule-find-perl
 libmodule-scandeps-perl libnet1 libn1-3-200 libproc-processtable-perl libprotobuf32
 libsort-naturally-perl libterm-readkey-perl libyaml-0-2 needrestart patch python3-attr python3-distro
 python3-distutils python3-docker python3-dockerpty python3-docopt python3-dotenv python3-json-pointer
 python3-jsonschema python3-lib2to3 python3-protobuf python3-pyrsistent python3-rfc3987 python3-texttable
 python3-uritemplate python3-webcolors python3-websocket python3-yaml runc sgml-base tini
Vorgeschlagene Pakete:
 binutils-doc containerd-networking-plugins docker-doc aufs-tools btrfs-progs debootstrap rinse rootlesskit
 xfsprogs zfs-fuse | zfsutils-linux git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk
 gitweb git-cvs git-mediawiki git-svn needrestart-session | libnotify-bin ed diffutils-doc python-attr-doc
 python-jsonschema-doc sgml-base-doc
Die folgenden NEUEN Pakete werden installiert:
 binutils binutils-common binutils-x86-64-linux-gnu cgroupfs-mount containerd criu docker-compose
 docker.io git git-man libbinutils libc6-nobfd0 libc60 liberror-perl libgprofng0 libintl-perl
 libintl-xs-perl libmodule-find-perl libmodule-scandeps-perl libnet1 libn1-3-200 libproc-processtable-perl
 libprotobuf32 libsort-naturally-perl libterm-readkey-perl libyaml-0-2 needrestart patch python3-attr
 python3-distro python3-distutils python3-docker python3-dockerpty python3-docopt python3-dotenv
 python3-json-pointer python3-jsonschema python3-lib2to3 python3-protobuf python3-pyrsistent
 python3-rfc3987 python3-texttable python3-uritemplate python3-webcolors python3-websocket python3-yaml
 runc sgml-base tini
0 aktualisiert, 49 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
Es müssen 84,8 MB an Archiven heruntergeladen werden.
Nach dieser Operation werden 361 MB Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren? [J/n] j
```

... werden die Pakete heruntergeladen, entpackt ...

```
wien.schubert.si x
Holen:32 http://deb.debian.org/debian bookworm/main amd64 python3-yaml amd64 6.0-3+b2 [119 kB]
Holen:33 http://deb.debian.org/debian bookworm/main amd64 docker-compose all 1.29.2-3 [123 kB]
Holen:34 http://deb.debian.org/debian bookworm/main amd64 liberror-perl all 0.17029-2 [29,0 kB]
Holen:35 http://deb.debian.org/debian bookworm/main amd64 git-man all 1:2.39.2-1.1 [2.049 kB]
Holen:36 http://deb.debian.org/debian bookworm/main amd64 git amd64 1:2.39.2-1.1 [7.171 kB]
Holen:37 http://deb.debian.org/debian bookworm/main amd64 libintl-perl all 1.33-1 [720 kB]
Holen:38 http://deb.debian.org/debian bookworm/main amd64 libintl-xs-perl amd64 1.33-1 [15,6 kB]
Holen:39 http://deb.debian.org/debian bookworm/main amd64 libmodule-find-perl all 0.16-2 [10,6 kB]
Holen:40 http://deb.debian.org/debian bookworm/main amd64 libmodule-scandeps-perl all 1.31-2 [41,7 kB]
Holen:41 http://deb.debian.org/debian bookworm/main amd64 libproc-processtable-perl amd64 0.634-1+b2 [43,1 kB]
]
Holen:42 http://deb.debian.org/debian bookworm/main amd64 libsort-naturally-perl all 1.03-4 [13,1 kB]
Holen:43 http://deb.debian.org/debian bookworm/main amd64 libterm-readkey-perl amd64 2.38-2+b1 [24,5 kB]
Holen:44 http://deb.debian.org/debian bookworm/main amd64 needrestart all 3.6-4+deb12u1 [59,8 kB]
Holen:45 http://deb.debian.org/debian bookworm/main amd64 patch amd64 2.7.6-7 [128 kB]
Holen:46 http://deb.debian.org/debian bookworm/main amd64 python3-json-pointer all 2.3-2 [15,1 kB]
Holen:47 http://deb.debian.org/debian bookworm/main amd64 python3-rfc3987 all 1.3.8-2 [8.816 B]
Holen:48 http://deb.debian.org/debian bookworm/main amd64 python3-uritemplate all 4.1.1-2 [10,9 kB]
Holen:49 http://deb.debian.org/debian bookworm/main amd64 python3-webcolors all 1.11.1-1 [12,7 kB]
Es wurden 84,8 MB in 2 s geholt (48,6 MB/s).
Extrahiere Vorlagen aus Paketen: 100%
Vormals nicht ausgewähltes Paket runc wird gewählt.
(Lese Datenbank ... 30057 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereitung zum Entpacken von .../00-runc_1.1.5+ds1-1+deb12u1_amd64.deb ...
Entpacken von runc (1.1.5+ds1-1+deb12u1) ...
Vormals nicht ausgewähltes Paket containerd wird gewählt.
Vorbereitung zum Entpacken von .../01-containerd_1.6.20~ds1-1+b1_amd64.deb ...
Entpacken von containerd (1.6.20~ds1-1+b1) ...

Fortschritt: [ 1%] [.....]
```


... und eingerichtet:

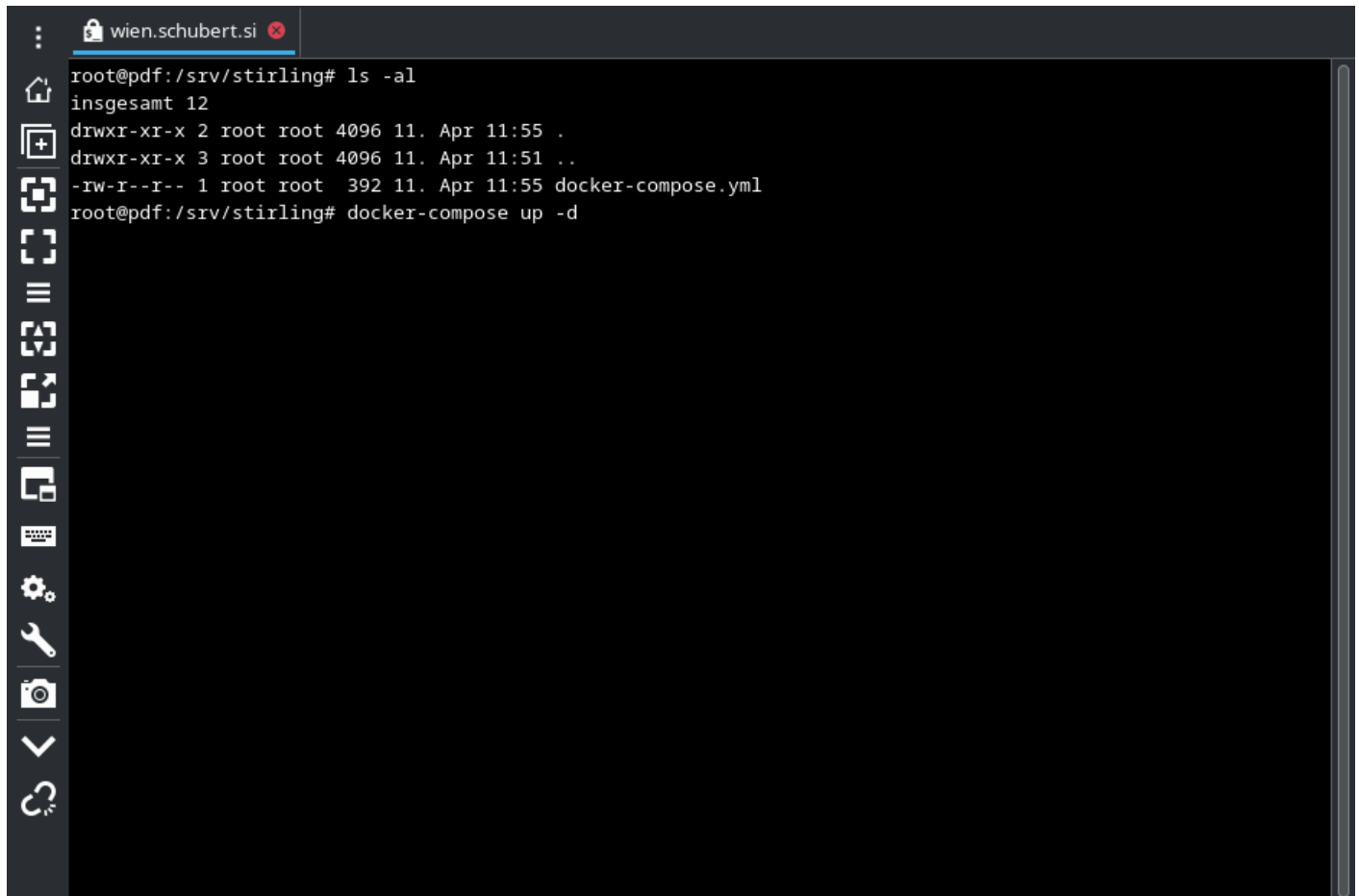


```
wien.schubert.si x
rd.service.
python3-lib2to3 (3.11.2-3) wird eingerichtet ...
libsort-naturally-perl (1.03-4) wird eingerichtet ...
python3-websocket (1.2.3-1) wird eingerichtet ...
libmodule-scandeps-perl (1.31-2) wird eingerichtet ...
python3-dockerpty (0.4.1-4) wird eingerichtet ...
libc64 (2.40-2) wird eingerichtet ...
python3-distutils (3.11.2-3) wird eingerichtet ...
python3-docker (5.0.3-1) wird eingerichtet ...
docker.io (20.10.24+dfsg1-1+b3) wird eingerichtet ...
Lege Gruppe »docker« (GID 109) an ...
Fertig.
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
libnet1:amd64 (1.1.6+dfsg-3.2) wird eingerichtet ...
libint1-xs-perl (1.33-1) wird eingerichtet ...
libprofng0:amd64 (2.40-2) wird eingerichtet ...
python3-jjsonschema (4.10.3-1) wird eingerichtet ...
git (1:2.39.2-1.1) wird eingerichtet ...
criu (3.17.1-2) wird eingerichtet ...
binutils-x86-64-linux-gnu (2.40-2) wird eingerichtet ...
docker-compose (1.29.2-3) wird eingerichtet ...
binutils (2.40-2) wird eingerichtet ...
needrestart (3.6-4+deb12u1) wird eingerichtet ...
Trigger für man-db (2.11.2-2) werden verarbeitet ...
Trigger für libc-bin (2.36-9+deb12u4) werden verarbeitet ...
root@pdf:/srv/stirling#
```

Docker und Docker-Compose sind jetzt betriebsbereit.

Docker-Container erstellen

In dem Verzeichnis, in dem auch die *docker-compose.yml* liegt, wird der Befehl `docker-compose up -d` abgesetzt:

A terminal window with a dark background and a light gray sidebar on the left containing various icons. The terminal title bar shows 'wien.schubert.si' with a red close button. The terminal text shows the user 'root' at 'pdf:/srv/stirling' running 'ls -al', which lists directory contents including 'docker-compose.yml'. Then, the user runs 'docker-compose up -d'.

```
root@pdf:/srv/stirling# ls -al
insgesamt 12
drwxr-xr-x 2 root root 4096 11. Apr 11:55 .
drwxr-xr-x 3 root root 4096 11. Apr 11:51 ..
-rw-r--r-- 1 root root  392 11. Apr 11:55 docker-compose.yml
root@pdf:/srv/stirling# docker-compose up -d
```

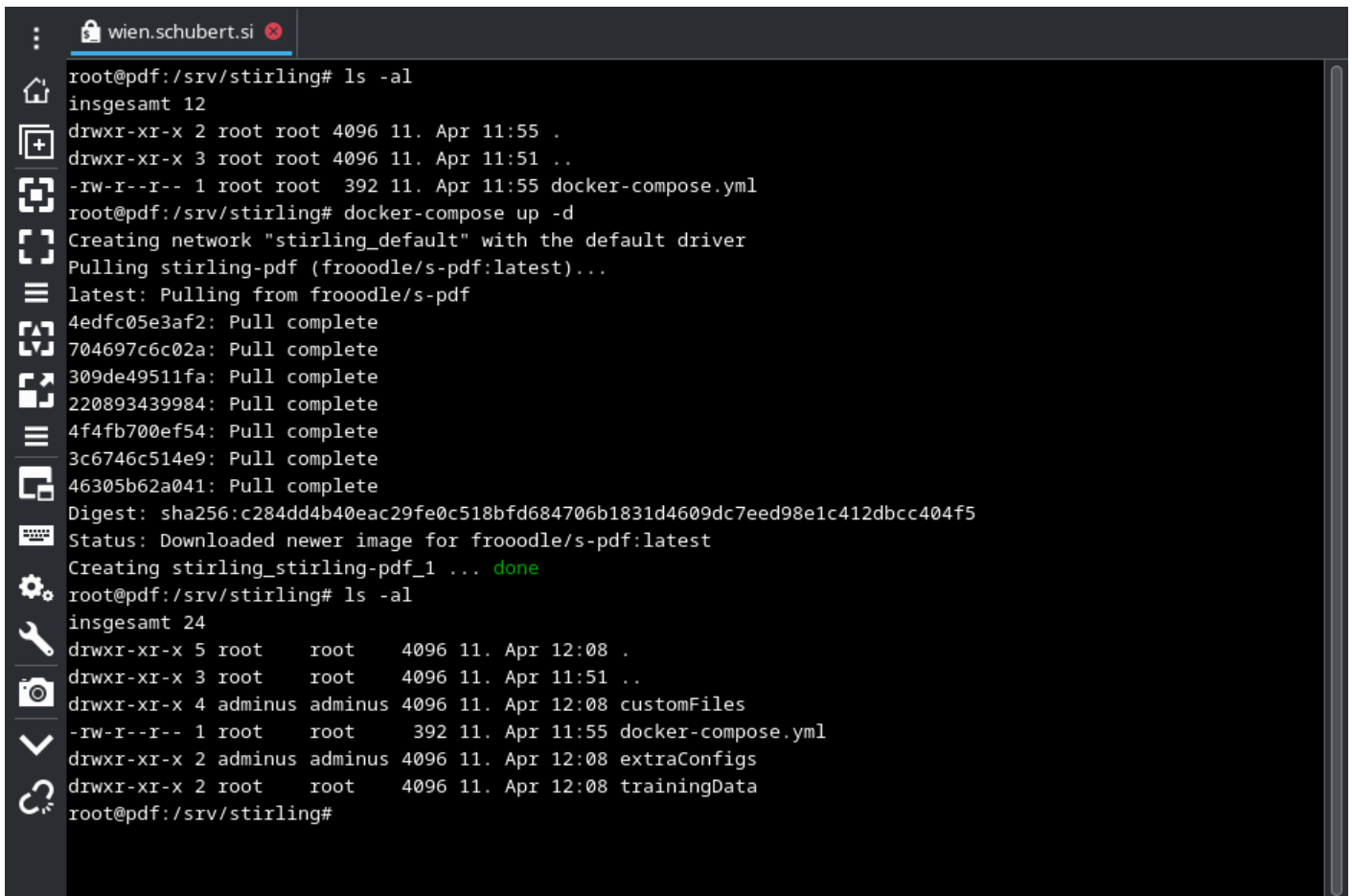
Daraufhin werden die benötigten Daten angefordert ...

```
wien.schubert.si x
root@pdf:/srv/stirling# ls -al
insgesamt 12
drwxr-xr-x 2 root root 4096 11. Apr 11:55 .
drwxr-xr-x 3 root root 4096 11. Apr 11:51 ..
-rw-r--r-- 1 root root 392 11. Apr 11:55 docker-compose.yml
root@pdf:/srv/stirling# docker-compose up -d
Creating network "stirling_default" with the default driver
Pulling stirling-pdf (frooodle/s-pdf:latest)...
latest: Pulling from frooodle/s-pdf
4edfc05e3af2: Pulling fs layer
704697c6c02a: Download complete
309de49511fa: Download complete
220893439984: Download complete
4f4fb700ef54: Download complete
3c6746c514e9: Download complete
46305b62a041: Downloading [=====>] 113.9MB/667.7MB
```

... ausgepackt ...

```
wien.schubert.si x
root@pdf:/srv/stirling# ls -al
insgesamt 12
drwxr-xr-x 2 root root 4096 11. Apr 11:55 .
drwxr-xr-x 3 root root 4096 11. Apr 11:51 ..
-rw-r--r-- 1 root root 392 11. Apr 11:55 docker-compose.yml
root@pdf:/srv/stirling# docker-compose up -d
Creating network "stirling_default" with the default driver
Pulling stirling-pdf (frooodle/s-pdf:latest)...
latest: Pulling from frooodle/s-pdf
4edfc05e3af2: Pull complete
704697c6c02a: Pull complete
309de49511fa: Pull complete
220893439984: Pull complete
4f4fb700ef54: Pull complete
3c6746c514e9: Pull complete
46305b62a041: Extracting [=====>] 349.3MB/667.7MB
```

... sowie der Container erstellt und in Betrieb genommen (Meldung **done**):

A terminal window with a dark background and a sidebar on the left containing various icons. The terminal shows the execution of Docker Compose commands to pull and start a container. The first 'ls -al' command shows 12 files. After running 'docker-compose up -d', the terminal shows the pulling of the 'stirling-pdf' image from 'frooodle/s-pdf:latest'. It lists several layers being pulled and their completion status. A digest is provided for the image. The status indicates that a newer image was downloaded. Finally, the container 'stirling_stirling-pdf_1' is created, and the word 'done' appears in green. A second 'ls -al' command shows 24 files, including 'customFiles', 'extraConfigs', and 'trainingData'.

```
wien.schubert.si x
root@pdf:/srv/stirling# ls -al
insgesamt 12
drwxr-xr-x 2 root root 4096 11. Apr 11:55 .
drwxr-xr-x 3 root root 4096 11. Apr 11:51 ..
-rw-r--r-- 1 root root 392 11. Apr 11:55 docker-compose.yml
root@pdf:/srv/stirling# docker-compose up -d
Creating network "stirling_default" with the default driver
Pulling stirring-pdf (frooodle/s-pdf:latest)...
latest: Pulling from frooodle/s-pdf
4edfc05e3af2: Pull complete
704697c6c02a: Pull complete
309de49511fa: Pull complete
220893439984: Pull complete
4f4fb700ef54: Pull complete
3c6746c514e9: Pull complete
46305b62a041: Pull complete
Digest: sha256:c284dd4b40eac29fe0c518bfd684706b1831d4609dc7eed98e1c412dbcc404f5
Status: Downloaded newer image for frooodle/s-pdf:latest
Creating stirring_stirling-pdf_1 ... done
root@pdf:/srv/stirling# ls -al
insgesamt 24
drwxr-xr-x 5 root root 4096 11. Apr 12:08 .
drwxr-xr-x 3 root root 4096 11. Apr 11:51 ..
drwxr-xr-x 4 adminus adminus 4096 11. Apr 12:08 customFiles
-rw-r--r-- 1 root root 392 11. Apr 11:55 docker-compose.yml
drwxr-xr-x 2 adminus adminus 4096 11. Apr 12:08 extraConfigs
drwxr-xr-x 2 root root 4096 11. Apr 12:08 trainingData
root@pdf:/srv/stirling#
```

Ab jetzt ist Stirling-PDF verfügbar.
(In diesem Beispiel via <http://IP-Adresse:8080>)

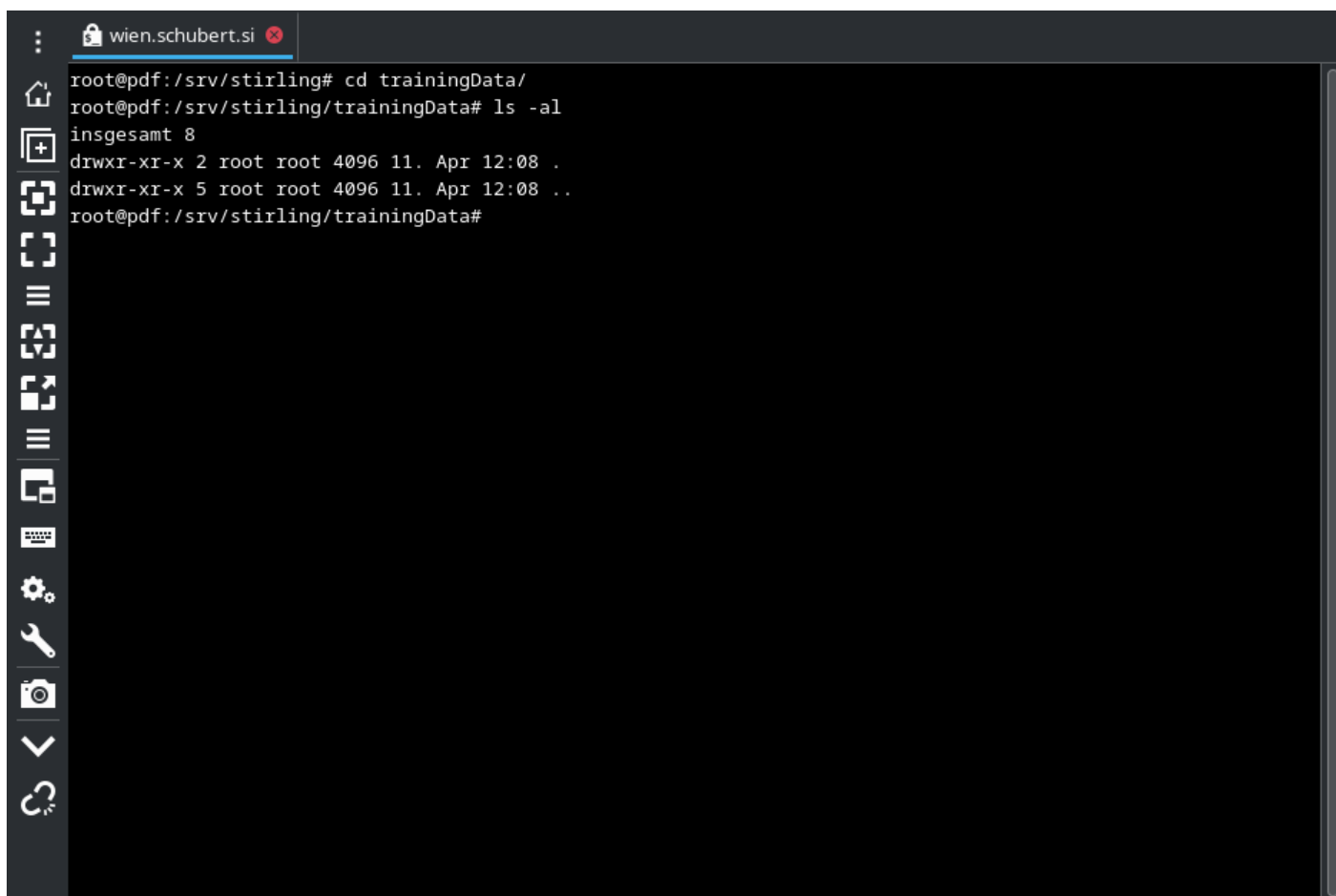
Sprachpakete hinzufügen

In der Grundinstallation verfügt Stirling-PDF nur über das englischsprachige Paket für die Texterkennung.

Um bspw. deutschsprachige Texte verarbeiten zu können, bedarf es somit weiterer Sprachpakete.

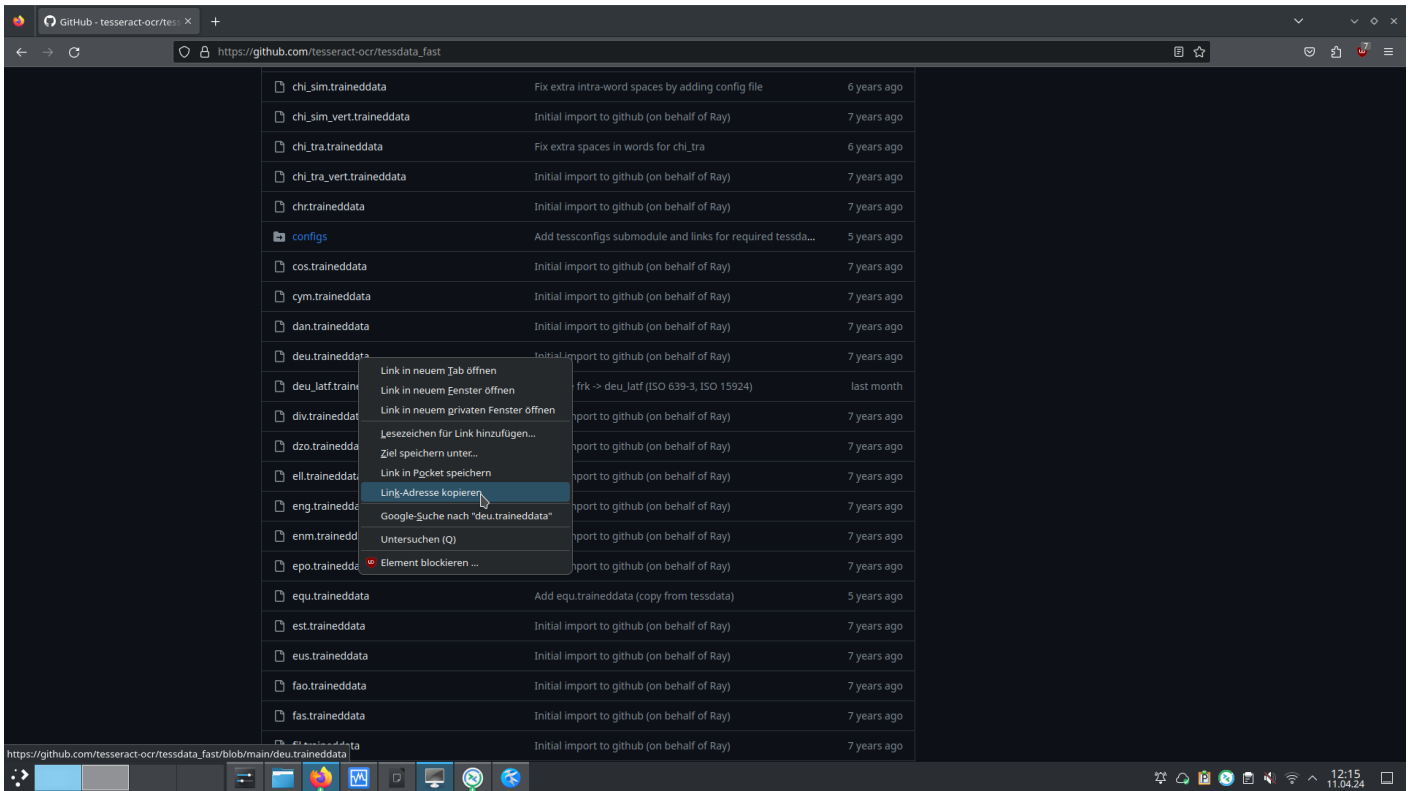
Die optionalen Sprachpakete befinden sich unterhalb des Installationsverzeichnis (hier */srv/stirling*) im Verzeichnis *trainingData*.

Daher wird zunächst in dieses Verzeichnis gewechselt, das nach der Installation noch leer ist:



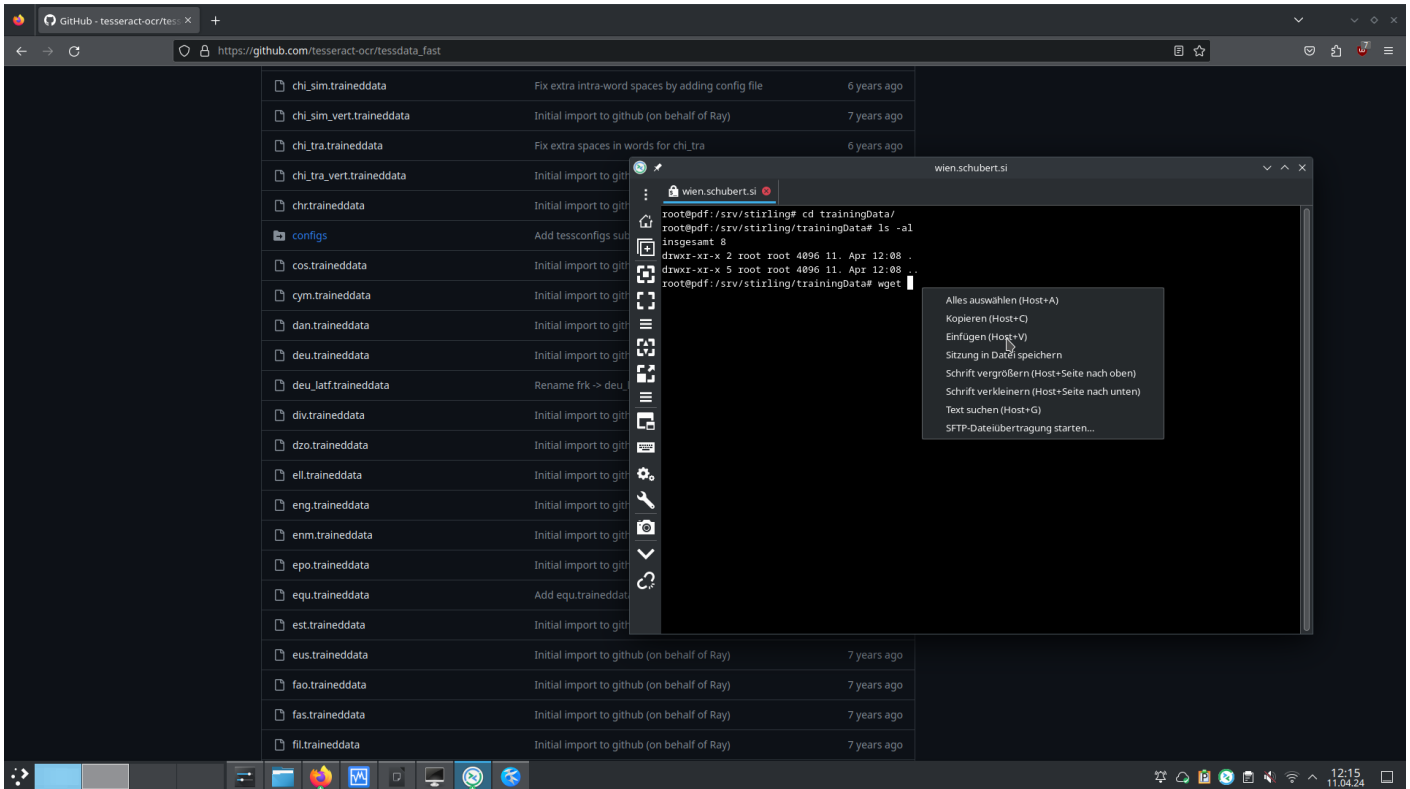
```
wien.schubert.si x
root@pdf:/srv/stirling# cd trainingData/
root@pdf:/srv/stirling/trainingData# ls -al
insgesamt 8
drwxr-xr-x 2 root root 4096 11. Apr 12:08 .
drwxr-xr-x 5 root root 4096 11. Apr 12:08 ..
root@pdf:/srv/stirling/trainingData#
```

Auf der [Übersichtsseite aller verfügbaren Sprachpakete](#) wird die URL des zu installierenden Paketes (in diesem Beispiel *deutsch*) ermittelt und kopiert:



Im Verzeichnis der Sprachpakete kann die gewünschte Datei dann mittels `wget URL` empfangen werden.

Für das Beispiel lautet der gesamte Befehl somit `wget https://github.com/tesseract-ocr/tessdata_fast/blob/main/deu.traineddata`.



Nach dem Download enthält das Verzeichnis *trainingData* die entsprechende Sprachdatei:

```
wien.schubert.si x
root@pdf:/srv/stirling/trainingData# ls -al
insgesamt 164
drwxr-xr-x 2 root root 4096 11. Apr 12:23 .
drwxr-xr-x 5 root root 4096 11. Apr 12:08 ..
-rw-r--r-- 1 root root 156627 11. Apr 12:23 deu.traineddata
root@pdf:/srv/stirling/trainingData#
```

Obwohl die Datei jetzt vorhanden ist, weiß die Anwendung (Stirling-PDF) noch nichts von ihr. Hierfür muss abschließend der Container neu gestartet werden.

Im ersten Schritt wird die laufende Anwendung per `docker-compose down` heruntergefahren und im zweiten Schritt durch `docker-compose up -d` wieder gestartet.

```
wien.schubert.si x
root@pdf:/srv/stirling/trainingData# docker-compose down
Removing stirling_stirling-pdf_1 ... done
Removing network stirling_default
root@pdf:/srv/stirling/trainingData# docker-compose up -d
Creating network "stirling_default" with the default driver
Creating stirling_stirling-pdf_1 ... done
root@pdf:/srv/stirling/trainingData#
```

Das zusätzliche Sprachpaket steht Stirling-PDF ab sofort zur Verfügung.

Bereitstellung von Stirling-PDF

Dieser Abschnitt beschreibt, wie Stirling-PDF normalerweise aufgerufen wird und wie sich dies optimieren lässt.

Zugang zur Anwendung

Out-of-the-Box ist Stirling-PDF nach dem Schema *HTTP://IP-ADRESSE:PortNummer* erreichbar,

konkret bedeutet dies bspw.:

bei IPv4 `http://1.2.3.4:8080`

bei IPv6 `http://[2:3:4:5:a:b:c:d]:8080`

Die Bereitstellung von Diensten per unverschlüsseltem HTTP ist unzeitgemäß und ein Sicherheitsrisiko.

Außerdem ist der Port 80 (Standard-Port für HTTP) häufig bereits durch einen (anderen) Webserver belegt und alternative Ports (hier 8080) können häufig aufgrund restriktiver Firewall-Regeln bspw. in Unternehmensnetzwerken nicht erreicht werden.

Ausgangslage und Zielsetzung

Das hier behandelte Szenario basiert auf folgender Infrastruktur:

- Zur Verfügung stehen eine [IPv4](#)-Adresse und ein [IPv6](#)-Subnet (Standard 64-Bit).
Die [IPv4](#)-Adresse ist einem [Proxmox](#)-Server zugewiesen, der diverse virtuelle Maschinen beherbergt, die sich in einem separaten Netzwerk ([Brücke mit NAT](#)) befinden.
Das [IPv6](#)-Subnet wird ebenfalls vom [Proxmox](#)-Server verwaltet und über [geroutete Brücken](#) an die virtuellen Maschinen verteilt.
- Eine dieser virtuellen Maschinen stellt einen Webserver [Apache](#) 2.4 als [ReverseProxy](#) zur Verfügung.
Der Traffic, der beim Proxmox-Host auf den Ports 80 (HTTP) und 443 (HTTPS) ankommt, wird direkt zu diesem [ReverseProxy](#) durchgeleitet.
- Stirling-PDF läuft auf einer weiteren virtuellen Maschinen.
- Die VM mit Stirling-PDF ist nicht direkt von außen zu erreichen:
Zwar könnte sie über eine IPv6-Adresse aus dem bereit gestellten Subnetz angesprochen werden -
aber damit wären all die Nutzer ausgeschlossen, die nur über eine IPv4-Anbindung verfügen.
Eine weitere (individuelle) IPv4-Adresse (nur für Stirling-PDF) scheidet aus Ressourcengründen (Kosten und Verfügbarkeit) aus.
Stirling-PDF wird über Port 8080 bereit gestellt -
dieser ist jedoch aus manchen Netzwerken nicht zu erreichen, weil insbesondere Unternehmensnetzwerke häufig sehr restriktive Firewall-Regeln haben und häufig nur HTTPS via Port 443 erlauben.
Eine Port-Weiterleitung von Port 80 am Host auf Port 8080 zur VM mit Stirling-PDF scheidet aus, weil Port 80 am Host bereits anderweitig belegt ist.

Ziel ist es, die Anwendung Stirling-PDF per sicherem HTTPS über den Standard-Port (443) zur Verfügung zu stellen.

Zugang per ReverseProxy

Die Konfiguration eines [ReverseProxy](#) und der Einsatz von [Let's Encrypt](#) werden in einem separaten Buch aufgegriffen.

Hier an dieser Stelle daher nur eine kurze Zusammenfassung.

Auf dem [ReverseProxy](#) (hier ein [Apache](#) 2.4 Web-Server) wird für dieses Beispiel folgende Konfiguration (bspw. als `/etc/apache2/sites-enabled/stirling.conf`) angelegt:

```
<VirtualHost subdomain.schubert.si:80>
    ServerAdmin webmaster@schubert.si
    ServerName subdomain.schubert.si
    ServerSignature Off
    DocumentRoot /var/www/html/stirling
    RewriteEngine on
    RewriteCond %{SERVER_NAME} =subdomain.schubert.si
    RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Das unter *DocumentRoot* angegebene Verzeichnis ist nur ein Dummy mit einer leeren *index.html*-Datei.

Gebraucht wird das leere Verzeichnis nur für den Einsatz des Certbot von [Let's Encrypt](#).

Auch wenn es zunächst so aussehen mag:

Es wird kein Dienst unter Port 80 (unverschlüsseltes HTTP) angeboten, sondern per Rewrite wird jeglicher Aufruf von HTTP auf HTTPS umgeschrieben.

Analog zur Konfiguration für Anfragen an Port 80 gibt es eine weitere Konfiguration (bspw. als `/etc/apache2/sites-enabled/stirling-le-ssl.conf`) oder als weiteren Abschnitt in der ersten Konfigurationsdatei, um eben die Anfragen an Port 443 (HTTPS) zu beantworten:

```
<IfModule mod_ssl.c>
<VirtualHost subdomain.schubert.si:443>
    ServerAdmin webmaster@schubert.si
    ServerName subdomain.schubert.si
```

```
ServerSignature Off
ProxyPreserveHost On
ProxyPass / http://192.168.1.11:8080/
ProxyPassReverse / http://192.168.1.11:8080/
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
SSLCertificateFile /etc/letsencrypt/live/subdomain.schubert.si/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/subdomain.schubert.si/privkey.pem
Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
</IfModule>
```

Hinter *ProxyPass*... wird die (interne) IPv4-Adresse der virtuellen Maschine angegeben, die Stirling-PDF bereitstellt.

Dafür müssen sich natürlich sowohl die VM mit dem [ReverseProxy](#) als auch die VM mit Stirling-PDF im gleichen Subnetz befinden.

Sollte der [ReverseProxy](#) auf der gleichen Maschine laufen wie Stirling-PDF selbst, wird stattdessen localhost (<http://localhost:8080> oder <http://127.0.0.1:8080>) verwendet

Nach Anlage der Konfiguration muss diese Apache noch mittels `systemctl reload apache2` bekannt gemacht werden.

Somit kann Stirling-PDF per HTTPS unter dem Standard-Port 443 anstatt über eine unverschlüsselte HTTP-Verbindung genutzt werden.