AUTOLOGBUCH

Installation auf dem Raspberry Pi

Kurzanleitung
für Einsteiger

Diese Anleitung führt Schritt für Schritt durch die Installation von AutoLogbuch auf einem eigenen Raspberry Pi. Anders als bei einer Synology-NAS gibt es dabei **kein grafisches Installationsprogramm** — ein paar Befehle werden über ein Terminal-Fenster eingegeben. Jeder Befehl steht hier vollständig zum Abtippen bzw. Kopieren bereit, Vorwissen ist nicht nötig. Der **Grundteil (A–F)** reicht aus, damit die App im eigenen WLAN nutzbar ist.

GRUNDINSTALLATION (A–F)
Nur im eigenen WLAN erreichbar. Ca. 45–60 Minuten (inkl. Wartezeiten). Kein Vorwissen nötig, aber ein zweiter Computer zum Vorbereiten der Speicherkarte/SSD.

OPTIONAL: SSD STATT SPEICHERKARTE, ZUGRIFF VON UNTERWEGS (G–H)
Schnellerer, zuverlässigerer Speicher bzw. Erreichbarkeit von jedem Ort. Setzt A–F voraus.

Welcher Raspberry Pi? — Empfehlung

MODELL
Raspberry Pi 5
4 GB RAM reichen für AutoLogbuch komfortabel aus (8 GB, falls Reserve für später gewünscht ist).

ZUBEHÖR
Netzteil (27 W USB-C), Gehäuse mit Lüfter, Netzwerkkabel oder WLAN
Ein aktiv gekühltes Gehäuse verhindert Leistungsdrosselung im Dauerbetrieb.

SPEICHER
Speicherkarte reicht zum Start
Für Dauerbetrieb empfohlen: NVMe-SSD statt Speicherkarte — siehe optionaler Teil G. Karten nutzen sich durch die Datenbank schneller ab.

AUCH MÖGLICH
Raspberry Pi 4 (4 GB)
Funktioniert ebenfalls, wenn schon vorhanden — spürbar langsamer beim einmaligen Bauen der App, SSD dann nur per USB statt eingebautem Steckplatz.

Das brauchst du zusätzlich

- Einen Windows-, Mac- oder Linux-Computer zum Vorbereiten der Speicherkarte/SSD und für die spätere Terminal-Verbindung
- Einen kostenlosen GitHub-Account (nur zum Herunterladen der App-Dateien, siehe Teil E)
- Beide Geräte im selben Netzwerk (WLAN oder Netzwerkkabel), mit Internetzugang

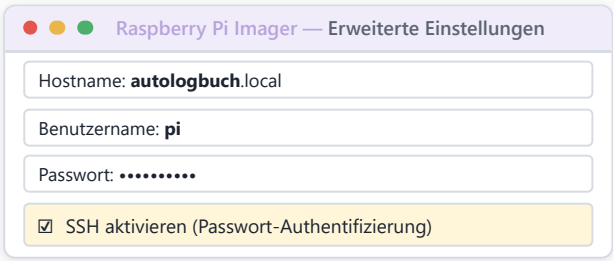
A Betriebssystem vorbereiten

1 Raspberry Pi Imager herunterladen und installieren
Auf dem zweiten Computer `raspberrypi.com/software` öffnen, **Raspberry Pi Imager** herunterladen und installieren. Speicherkarte (bzw. die SSD, siehe Teil G) per Kartenleser/USB-Adapter an diesen Computer anschließen.

2 Gerät, Betriebssystem und Speicher auswählen
Imager öffnen → **Gerät wählen**: Raspberry Pi 5 (bzw. 4). **Betriebssystem wählen** → *Raspberry Pi OS (other)* → **Raspberry Pi OS Lite (64-bit)** — ohne grafische Oberfläche, für den Dauerbetrieb genau richtig. **Speicher wählen**: die angeschlossene Karte/SSD.

3 Zugangsdaten schon jetzt festlegen (wichtig!)
Vor dem Schreiben auf ⚙ **Einstellungen bearbeiten** klicken. Dort: **Hostname** vergeben (z. B. `autologbuch`), **Benutzername und Passwort festlegen** ausfüllen, unter **Dienste** den Haken bei **SSH aktivieren** setzen (Passwort-Authentifizierung), optional WLAN-Zugangsdaten eintragen (bei Netzwerkkabel nicht nötig). Danach **Speichern** und **Schreiben** klicken.

Diese Einstellungen ermöglichen die spätere Verbindung per Terminal, ganz ohne Bildschirm/Tastatur am Raspberry Pi selbst anzuschließen.



Merken!
Hostname, Benutzername und Passwort werden im nächsten Teil für die Verbindung gebraucht — am besten kurz notieren.

B Ersten Start durchführen

4 Karte/SSD einsetzen und einschalten

Speicherkarte bzw. SSD in den Raspberry Pi einsetzen, Netzwerkkabel anschließen (empfohlen) oder auf WLAN verlassen, Netzteil einstecken. Der erste Start dauert **2–3 Minuten** (die vorbereiteten Einstellungen werden dabei übernommen).

C Per Terminal verbinden (SSH)

5 Terminal öffnen

Windows: Startmenü → „PowerShell" eingeben und öffnen. **Mac**: „Terminal" über die Spotlight-Suche öffnen. Beide bringen die nötige SSH-Funktion bereits mit, es muss nichts zusätzlich installiert werden.

6 Verbinden

Den Befehl aus dem Terminal-Fenster rechts eintippen (Hostname bzw. Benutzername aus Teil A, Schritt 3 einsetzen) und mit **Eingabe** bestätigen. Bei der Frage nach dem „Fingerprint" mit `yes` bestätigen (nur beim allerersten Mal), danach das vorher festgelegte Passwort eingeben — **beim Tippen erscheint bewusst kein Sternchen**, das ist normal.

Meldet sich `autologbuch.local` nicht? Dann stattdessen die IP-Adresse verwenden — zu finden in der Geräteliste des Routers (meist unter „Heimnetz" oder „Netzwerkübersicht").

PowerShell

```
PS> ssh pi@autologbuch.local
The authenticity of host 'autologbuch.local' can't
be established.
Are you sure you want to continue connecting
(yes/no)? yes
pi@autologbuch.local's password:
// Passwort eintippen (unsichtbar) + Eingabe
pi@autologbuch:~$
```

D System aktualisieren und Docker installieren

7 System auf den neuesten Stand bringen

Im bereits geöffneten Terminal (jetzt mit dem Raspberry Pi verbunden) die erste Zeile rechts eintippen und mit Eingabe bestätigen. Dauert **2–5 Minuten**.

8 Docker installieren

Danach die zweite und dritte Zeile eintippen — das offizielle Installationsskript von Docker richtet Docker **und** Docker Compose in einem Schritt ein. Dauert **3–5 Minuten**.

9 Einmal neu starten

Die letzte Zeile startet den Raspberry Pi neu, damit die Docker-Berechtigung wirksam wird. Die Verbindung im Terminal bricht dabei ab — das ist normal. Nach **ca. 1 Minute** mit demselben Befehl aus Teil C, Schritt 6 erneut verbinden.

pi@autologbuch

```
$ sudo apt update && sudo apt upgrade -y
$ curl -fsSL https://get.docker.com -o get-
docker.sh
$ sudo sh get-docker.sh
$ sudo usermod -aG docker $USER
$ sudo reboot
```

Was passiert hier?

Diese vier Zeilen entsprechen genau dem Öffnen des „Paketzentrums" bei einer Synology-NAS — nur eben als Text statt als Klick. Docker wird damit dauerhaft installiert, kein erneutes Ausführen nötig.

E App-Dateien laden

10 Git installieren

Im (neu verbundenen) Terminal die erste Zeile eintippen — Git wird gebraucht, um die App-Dateien von GitHub herunterzuladen.

11 App-Dateien herunterladen

Zweite Zeile eintippen (der Zugriffstoken ist bereits enthalten). Legt den Ordner `autoapp` im Home-Verzeichnis automatisch mit allen App-Dateien an. Danach mit der dritten Zeile in den neuen Ordner wechseln — **ab jetzt bleiben alle weiteren Befehle in diesem Ordner**.

```
pi@autologbuch
$ sudo apt install -y git
$ git clone https://github_pat_11AB…
@github.com/…/AutoLogbuch.git autoapp
$ cd autoapp
```

F Zugangsdaten eintragen (.env-Datei)

12 Vorlage kopieren

Erste Zeile rechts eintippen — legt eine Kopie namens `.env` an, die eigentliche Vorlage `.env.example` bleibt unverändert erhalten.

13 Datei mit dem eingebauten Editor öffnen und bearbeiten

Zweite Zeile eintippen — öffnet den einfachen Text-Editor `nano` direkt im Terminal. Mit den **Pfeiltasten** zu den zwei folgenden Zeilen navigieren und die Werte ändern, alles andere unverändert lassen:

`POSTGRES_PASSWORD` – frei erfundenes Passwort, mind. 12 Zeichen

`PUBLIC_APP_URL` – vorerst `http://raspberrypi-IP:8085`

Stehen bereits Zeilen wie `SMTP_…`, `ANTHROPIC_API_KEY=…` oder `FEEDBACK_SYNC_ROLE=…` mit einem Wert darin? Dann nichts ändern — E-Mail-Versand, die optionale KI-Felderkennung und ggf. die Verbindung zu einer zentralen App sind bereits fertig eingerichtet. Soll diese Verbindung erst noch eingerichtet werden: optionaler **Teil K** auf den letzten Seiten.

14 Speichern und schließen

`Strg+O` speichert (danach **Eingabe** bestätigen, der Dateiname bleibt unverändert), `Strg+X` schließt den Editor wieder.

```
pi@autologbuch — ~/autoapp
$ cp .env.example .env
$ nano .env
```

```
nano — .env
1  POSTGRES_USER=autoapp
2  POSTGRES_PASSWORD=MeinSicheres-Passwort9
3  POSTGRES_DB=autoapp
4  …
9  PUBLIC_APP_URL=http://192.168.1.60:8085
^O Speichern   ^X Beenden
```

G App bauen und starten

15 Ersten Bauvorgang anstoßen

Im Ordner `autoapp` (siehe Teil E) den Befehl rechts eintippen. Baut Datenbank, Backend und Oberfläche und startet alles automatisch. Dauert beim allerersten Mal **15–30 Minuten** — der Raspberry Pi ist dabei spürbar ausgelastet, das ist normal. Das Terminal bleibt währenddessen mit Textausgaben beschäftigt, bis am Ende wieder die normale Eingabezeile erscheint.

16 Status prüfen

Mit dem zweiten Befehl den Status aller drei Bausteine prüfen — bei allen drei sollte `running` bzw. `healthy` stehen.

```
pi@autologbuch — ~/autoapp
$ docker compose up -d --build
// … mehrere Minuten Textausgabe …
$ docker compose ps
postgres running (healthy)
backend running
frontend running
```

H App öffnen und einrichten

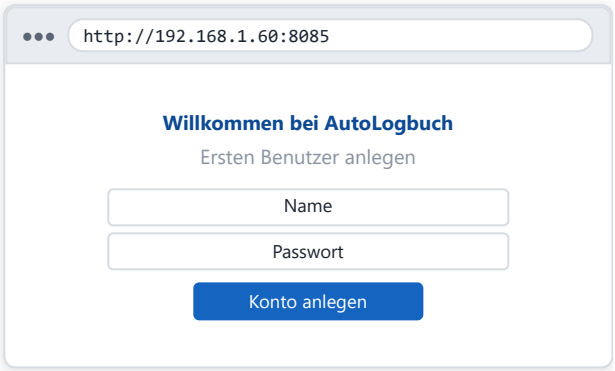
17 IP-Adresse des Raspberry Pi notieren

Mit dem Befehl rechts anzeigen lassen, oder direkt in der Geräteliste des Routers nachsehen. Alternativ funktioniert vom selben Computer aus auch der Hostname aus Teil A, z. B. `http://autologbuch.local:8085`.

18 Adresse im Browser öffnen

Auf jedem Gerät im selben WLAN: `http://raspberrypi-IP:8085` öffnen. Der Einrichtungs-Assistent legt beim allerersten Aufruf den ersten Benutzer an — **fertig!**

```
pi@autologbuch
$ hostname -I
192.168.1.60
```



Grundinstallation abgeschlossen

Die App ist jetzt im eigenen WLAN nutzbar. Für schnelleren/zuverlässigeren Speicher weiter mit Teil I, für Zugriff von unterwegs mit Teil J.

Testphase

Ohne Lizenzschlüssel ist die App ab jetzt 30 Tage lang voll nutzbar. Danach kann nur noch gelesen werden, bis der erste angelegte Benutzer unter **Mein Konto** einen Lizenzschlüssel einträgt — dafür bei der Person nachfragen, die diese Anleitung zur Verfügung gestellt hat.

I Optional: NVMe-SSD statt Speicherkarte

Empfohlen für den Dauerbetrieb: Speicherkarten nutzen sich durch die vielen kleinen Schreibzugriffe der Datenbank spürbar schneller ab als bei normaler Nutzung und können nach 1–2 Jahren Fehler verursachen. Eine NVMe-SSD ist zusätzlich 5–10× schneller.

I1 Passendes Zubehör besorgen

Nur beim Raspberry Pi 5 direkt möglich: ein **M.2-HAT+** (offizielles Zubehör oder kompatibles Fremdprodukt) plus eine **NVMe-SSD** (128–256 GB reichen sehr großzügig, ca. 20–25 €). Der HAT wird laut beiliegender Anleitung auf die Unterseite des Raspberry Pi geschraubt, die SSD wird dort eingesteckt.

I2 Direkt auf die SSD schreiben statt auf eine Speicherkarte

Genau wie in Teil A: die SSD stattdessen über einen USB-zu-M.2-Adapter an den Vorbereitungs-Computer anschließen, im Raspberry Pi Imager als Ziel-Speicher auswählen, dieselben Einstellungen (Schritt 3) vergeben. Danach die SSD in den HAT einsetzen statt eine Speicherkarte einzulegen — der Rest der Anleitung (Teil B–H) bleibt unverändert.

Bereits mit Speicherkarte installiert?

Kein Problem — einfach die SSD wie oben separat vorbereiten (dabei entsteht eine komplett neue, leere Installation) und danach Teil E–H auf der SSD wiederholen. Ein nachträgliches Umziehen der bestehenden Daten ist möglich, aber ein größerer Schritt — im Zweifel bei der Person nachfragen, die diese Anleitung zur Verfügung gestellt hat.

J Optional: Zugriff von unterwegs einrichten

Nur nötig, wenn die App auch außerhalb des eigenen WLANs erreichbar sein soll. Ein Raspberry Pi bringt anders als eine Synology-NAS keinen eingebauten Reverse-Proxy mit — dafür kommt hier **Caddy** zum Einsatz, ein zusätzlicher, sehr einfach einzurichtender Baustein, der sich automatisch um ein kostenloses Sicherheitszertifikat kümmert.

J1 Kostenlose Adresse (DDNS) einrichten

`duckdns.org` öffnen, mit einem bestehenden Konto (z. B. Google) anmelden, einen frei wählbaren Namen anlegen (z. B. `meinefamilie` → `meinefamilie.duckdns.org`) — diese Adresse zeigt ab sofort immer auf den eigenen Internetanschluss, auch wenn sich dessen Adresse ändert. Den angezeigten **Token** notieren.

J2 Caddy als weiteren Baustein ergänzen

Im Ordner `autoapp` eine neue Datei `Caddyfile` anlegen (`nano Caddyfile`) mit dem Inhalt rechts, dabei `meinefamilie.duckdns.org` durch die eigene Adresse aus J1 ersetzen. Speichern mit `Strg+O`, schließen mit `Strg+X`.

nano — Caddyfile

```
meinefamilie.duckdns.org {
  reverse_proxy frontend:80
}
```

J3 Caddy in die docker-compose.yml eintragen

`nano docker-compose.yml` öffnen, ganz unten bei `services:` den Block rechts einfügen (gleiche Einrückung wie `frontend:` darüber) und unter `volumes:` ganz unten die Zeile `caddy-data:` ergänzen. Speichern, schließen.

nano — docker-compose.yml

```
  caddy:
    image: caddy:2-alpine
    restart: unless-stopped
    ports:
      - '443:443'
    volumes:
      - ./Caddyfile:/etc/caddy/Caddyfile
      - caddy-data:/data
```

J4 Portweiterleitung im Router einrichten

Der Router muss Anfragen von außen an **Port 443** an die interne IP-Adresse des Raspberry Pi weiterreichen. Beispiel für eine **FritzBox**: `fritz.box` öffnen → **Internet** → **Freigaben** → **Portfreigaben** → **Gerät für Freigaben hinzufügen** → Raspberry Pi auswählen → Bezeichnung „AutoLogbuch", Protokoll **TCP**, Port außen und innen jeweils `443` → **OK**.

J5 Adresse anpassen und neu starten

In der `.env` (Teil F) `PUBLIC_APP_URL` auf `https://meinefamilie.duckdns.org` ändern, danach:

pi@autologbuch — ~/autoapp

```
$ docker compose up -d --build
```

Caddy besorgt sich beim ersten Start automatisch ein kostenloses Sicherheitszertifikat (Let's Encrypt) für die eigene Adresse — kein zusätzlicher Schritt nötig. Die App ist danach zusätzlich von überall über `https://meinefamilie.duckdns.org` erreichbar.

K Optional: Verbindung zu einer zentralen Installation (Feedback-Synchronisation)

Nur nötig, wenn über **„Feedback einreichen"** eingereichte Meldungen dieser Installation automatisch an eine bereits bestehende ANDERE AutoLogbuch-Installation (die „Zentrale") weitergeleitet werden sollen. Nur eine Installation im Einsatz: dieser Teil kann übersprungen werden.

K1 Zugangs-Token auf der Zentrale erzeugen

Auf der ANDEREN, bereits bestehenden Installation (der Zentrale) als Admin einloggen → **Einstellungen** → **Installationen** → **Neue Installation**, einen Namen vergeben. Der Zugangs-Token wird nur EINMAL im Klartext angezeigt — sicher an die Person weitergeben, die DIESEN Raspberry Pi hier betreibt.

pi@autologbuch — ~/autoapp

```
$ docker compose up -d --build
```

K2 Zugangsdaten in dieser Installation eintragen

Wie in Teil F: `nano .env` öffnen, folgende drei Zeilen eintragen bzw. anpassen:

```
FEEDBACK_SYNC_ROLE = satellite
FEEDBACK_CENTRAL_URL – Adresse der Zentrale
FEEDBACK_CENTRAL_TOKEN – der in K1 erzeugte Token
```

Diese Installation soll selbst die Zentrale sein?
Dann `FEEDBACK_SYNC_ROLE=central` statt `satellite` setzen (K2 sonst entfällt) — jede WEITERE Installation durchläuft dann K1–K3 mit DIESER Adresse als Zentrale.

K3 Neu starten

Wie im Terminal-Fenster gezeigt: `docker compose up -d --build` erneut ausführen. Ab jetzt werden eingereichte Meldungen automatisch (alle 15 Minuten) an die Zentrale übertragen, deren Bearbeitungsstatus wird ebenfalls automatisch zurückgemeldet. Ohne diese drei Zeilen funktioniert „Feedback einreichen" weiterhin ganz normal, nur rein lokal — dieser Teil ist jederzeit nachträglich einrichtbar.

? Häufige Fragen

„ssh: Could not resolve hostname" beim Verbinden. Raspberry Pi eingeschaltet und im selben Netzwerk? Alternativ die IP-Adresse aus der Geräteliste des Routers statt des Hostnamens verwenden.
Ein Baustein steht nicht auf „running"/„healthy". Mit `docker compose logs backend` (bzw. `postgres` / `frontend`) die Fehlermeldung ansehen — meist ein falsch eingetragener Wert in der `.env` (Teil F), danach `docker compose up -d --build` erneut ausführen.
Die Passwort-vergessen-E-Mail kommt nicht an. Der E-Mail-Versand ist bereits fertig eingerichtet (Teil F) — dafür bei der Person nachfragen, die diese Anleitung zur Verfügung gestellt hat. **Teil E, Schritt 11 lädt nichts herunter?** Der enthaltene Zugriffstoken kann abgelaufen sein — ebenfalls bei dieser Person nachfragen, sie kann einen neuen ausstellen. **Nach 30 Tagen nur noch Lesezugriff?** Testphase abgelaufen — Lizenzschlüssel bei dieser Person anfragen.
Updates erhalten. Im Ordner `autoapp` per Terminal: `git pull && docker compose up -d --build` — deine Daten (Fahrzeuge, Einträge) bleiben unberührt.