Die /boot/firmware/user-data Datei in Raspberry Pi OS Trixie

Automatisierte Systemkonfiguration mit cloud-init

.

Inhaltsverzeichnis

1. Einführung
2. Zweck und Funktionsweise
3. Typische Anwendungen
4. Konfigurationsoptionen
5. Vollständiges Beispiel
6. Debugging
7. Wichtige Hinweise

.

Einführung

Die `user-data`-Datei im Verzeichnis `/boot/firmware/` ist ein zentraler Bestandteil des **cloud-init**-Systems, das in Raspberry Pi OS (seit Bookworm und Trixie) zur automatischen Erstkonsfiguration verwendet wird.

Cloud-init ermöglicht die **automatisierte Konfiguration beim ersten Boot** eines Raspberry Pi, ohne dass eine interaktive Einrichtung erforderlich ist.

.

Zweck und Funktionsweise

Warum cloud-init?

Cloud-init wurde entwickelt, um die Bereitstellung von Systemen zu automatisieren. Auf dem Raspberry Pi ist dies besonders nützlich für:

- **Headless-Setups**: Installation ohne Monitor und Tastatur
- **Massenbereitstellung**: Viele Raspberry Pis identisch konfigurieren
- **Reproduzierbare Installationen**: Immer gleiche Konfiguration sicherstellen
- **Remote-Deployment**: SD-Karte vorbereiten, einlegen, fertig

Funktionsweise

Schritt	Beschreibung
1. Vorbereitung	`user-data` und `meta-data` auf die Boot-Partition kopieren
2. Erster Boot	Raspberry Pi startet erstmals
3. cloud-init läuft	System liest die Konfigurationsdateien
4. Anwendung	Alle Einstellungen werden automatisch übernommen
5. Abschluss	cloud-init deaktiviert sich für folgende Boots

Dateistruktur

```
/boot/firmware/
├── user-data      # Hauptkonfiguration (YAML-Format)
└── meta-data      # Metadaten (muss existieren, kann leer sein)
```

Wichtig: Die `meta-data`-Datei **muss existieren**, auch wenn sie leer ist:

```
touch /boot/firmware/meta-data
```

.

Typische Anwendungen

1. SSH-Zugang einrichten

```
#cloud-config
users:
  - name: pi
    ssh_authorized_keys:
      - ssh-rsa AAAAB3NzaC1yc2EAAAADAQAB... public_key
    sudo: ALL=(ALL) NOPASSWD:ALL
    shell: /bin/bash
```

2. Hostname setzen

```
#cloud-config
hostname: mein-raspberry-pi
```

3. Netzwerkkonfiguration (statisch)

```
#cloud-config
network:
  version: 2
  ethernets:
    eth0:
      addresses: [192.168.1.100/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]
```

4. WLAN konfigurieren

```
#cloud-config
network:
  version: 2
  wifis:
    wlan0:
      dhcp4: true
      access-points:
        "MeinNetzwerk":
          password: "MeinPasswort123"
```

5. Pakete installieren

```
#cloud-config
packages:
  - vim
  - git
  - htop
  - curl
  - wget
```

6. Beliebige Befehle ausführen

```
#cloud-config
runcmd:
  - apt update
  - apt upgrade -y
  - systemctl enable ssh
  - systemctl start ssh
  - echo "Setup abgeschlossen" >> /var/log/setup.log
```

.

Konfigurationsoptionen

Übersicht der wichtigsten Optionen

Option	Typ	Beschreibung
`hostname`	String	Setzt den Systemnamen
`timezone`	String	Setzt die Zeitzone (z.B. Europe/Berlin)
`locale`	String	Konfiguriert die Systemsprache

Option	Typ	Beschreibung
`users`	Liste	Legt Benutzer an, verwaltet SSH-Keys
`network`	Objekt	Netzwerkkonfiguration (Version 2)
`packages`	Liste	Installiert Pakete beim ersten Boot
`runcmd`	Liste	Führt Shell-Befehle aus
`write_files`	Liste	Schreibt Dateien ins System
`disable_root`	Boolean	Deaktiviert root-Login

Benutzer konfigurieren (users)

```yaml
users:
  - name: admin
    groups: sudo, docker
    shell: /bin/bash
    ssh_authorized_keys:
      - ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAA... user@host
    sudo: ALL=(ALL) NOPASSWD:ALL
    passwd: $6$rounds=4096$...  # verschlüsseltes Passwort
```

Dateien schreiben (write_files)

```yaml
write_files:
  - path: /etc/motd
    content: |
      Willkommen auf meinem Raspberry Pi!
    permissions: '0644'

  - path: /opt/mein-script.sh
    content: |
      #!/bin/bash
      echo "Hallo Welt"
    permissions: '0755'
```

.

Vollständiges Beispiel

Komplette Headless-Konfiguration

```yaml
#cloud-config
# Raspberry Pi OS Trixie - Vollkonfiguration

# System-Einstellungen
hostname: pi-server
timezone: Europe/Berlin
locale: de_DE.UTF-8

# Benutzer konfigurieren
users:
  - name: admin
    groups: sudo, docker
    shell: /bin/bash
    ssh_authorized_keys:
      - ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAI... admin@example.com
    sudo: ALL=(ALL) NOPASSWD:ALL

# WLAN konfigurieren
network:
  version: 2
  wifis:
    wlan0:
      dhcp4: true
      access-points:
        "MeinHeimnetz":
          password: "SicheresPasswort123"

# Pakete installieren
packages:
  - vim
  - git
  - curl
  - wget
```

```
  - htop
  - iotop
  - ncdu
  - python3-pip

# Befehle beim ersten Boot ausführen
runcmd:
  - echo "=== Starte Setup ===" >> /var/log/pi-setup.log
  - apt update >> /var/log/pi-setup.log 2>&1
  - apt upgrade -y >> /var/log/pi-setup.log 2>&1
  - systemctl enable ssh
  - systemctl start ssh
  - echo "=== Setup abgeschlossen ===" >> /var/log/pi-setup.log

# Abschlussnachricht
final_message: "Das System ist nach $UPTIME Sekunden einsatzbereit!"
```

.

Debugging

Logs prüfen

```
# Haupt-Logdatei
sudo cat /var/log/cloud-init.log

# Ausgabe der ausgeführten Befehle
sudo cat /var/log/cloud-init-output.log

# Status abfragen
sudo cloud-init status

# Detaillierter Status
sudo cloud-init status --long
```

cloud-init neu ausführen (nur zu Testzwecken!)

```
# cloud-init zurücksetzen
sudo cloud-init clean --logs

# Neu initialisieren
sudo cloud-init init

# Konfiguration anwenden
sudo cloud-init modules --mode=config

# Alles neu ausführen
sudo cloud-init init --local
sudo cloud-init init
```

Häufige Probleme

Problem	Lösung
cloud-init läuft nicht	`meta-data`-Datei fehlt
YAML-Fehler	Einrückungen prüfen (keine Tabs!)
Netzwerk nicht aktiv	`network:` Syntax prüfen
SSH-Keys nicht erkannt	Keys müssen in einer Zeile sein
Befehle nicht ausgeführt	`runcmd:` als Liste formatieren

YAML-Validierung

```
# YAML-Syntax prüfen
python3 -c "import yaml; yaml.safe_load(open('/boot/firmware/user-data'))"
```

.

Wichtige Hinweise

⚠ Achtung

Hinweis	Beschreibung
Nur beim ersten Boot	cloud-init läuft standardmäßig nur einmal
meta-data erforderlich	Datei muss existieren (kann leer sein)
YAML-Syntax	Korrekte Einrückungen sind kritisch
Keine Tabs verwenden	Immer Leerzeichen für Einrückungen
Backup erstellen	Vor Änderungen die SD-Karte sichern

Unterschied zu früheren Versionen

Raspberry Pi OS	Konfigurationsmethode
Buster/Bullseye	`ssh`-Datei + `wpa_supplicant.conf`
Bookworm/Trixie	`user-data` + `meta-data` (cloud-init)

Vorteile von cloud-init

Vorteil	Beschreibung
Industriestandard	Gleiche Syntax wie AWS, Azure, Google Cloud
Umfangreich	Viel mehr Optionen als die alte Methode
Gut dokumentiert	Umfangreiche offizielle Dokumentation
Getestet	Bewährt in Millionen von Installationen
Flexibel	Von einfach bis komplex skalierbar

Best Practices

1. **SSH-Keys statt Passwörtern**: Immer SSH-Keys verwenden
2. **Minimale Konfiguration**: Nur notwendige Einstellungen aufnehmen
3. **Testen**: Konfiguration erst in VM testen
4. **Versionierung**: user-data in Git speichern
5. **Dokumentation**: Kommentare im YAML hinzufügen

.

Nützliche Links

- **Offizielle cloud-init Dokumentation**: https://cloudinit.readthedocs.io/
- **Raspberry Pi Dokumentation**: https://www.raspberrypi.com/documentation/
- **cloud-init Beispiele**: https://cloudinit.readthedocs.io/en/latest/reference/examples.html
- **YAML Validator**: https://www.yamllint.com/

.

Zusammenfassung

Die `/boot/firmware/user-data`-Datei ist das moderne Werkzeug zur automatischen Konfiguration von Raspberry Pi Systemen unter Raspberry Pi OS Trixie. Sie ersetzt die veraltete Methode mit `ssh`-Datei und `wpa_supplicant.conf` und bietet:

- ✓ **Einheitliches Format** (YAML)
- ✓ **Mehr Funktionen** (Benutzer, Netzwerk, Pakete, Skripte)

- ✓ **Industriestandard** (cloud-init)
- ✓ **Bessere Wartbarkeit**
- ✓ **Ideal für Headless-Setups**

Empfehlung: Für jede neue Raspberry Pi-Installation sollte eine `user-data`-Datei vorbereitet werden, um Zeit zu sparen und Konsistenz sicherzustellen.

.

Erstellt am: 6. März 2026 Für Raspberry Pi OS Trixie mit cloud-init