Raspberry Pi 5 Server — Brugervejledning

Systemdokumentation

28. marts 2026

Indhold

Systemoverblik

Denne Raspberry Pi 5 kører som en lokal webserver med følgende tjenester:

Tjeneste	Adresse	Beskrivelse
Webserver	https://192.168.0.200	Nginx + PHP 8.4
phpMyAdmin	https://192.168.0.200/phpmyadmin/	Database-administration
Flask Dashboard	https://dashboard.local/	Python dashboard
MQTT Broker	192.168.0.200:1883	Mosquitto
MQTT WebSockets	192.168.0.200:9001	Mosquitto WS
SSH	192.168.0.200:22	Fjernadgang

Fast IP-adresse: 192.168.0.200 **Hostname:** raspberrypi / raspberrypi.local

Webserver — Nginx

Hvad er Nginx?

Nginx er webserveren der modtager alle indgående HTTP/HTTPS-forespørgsler og sender dem videre til den rigtige tjeneste — enten PHP-FPM (for PHP-sider) eller Flask (for dashboard).

Adgang

Åbn en browser og gå til:

`https://192.168.0.200/`

Velkomstsiden viser live status på alle tjenester.

Tilføj en ny PHP-side

Opret en fil i webrooten:

```
sudo nano /var/www/html/minside.php
```

Eksempel på en simpel PHP-side:

```
<?php
echo "<h1>Hej fra RPI!</h1>";
echo "<p>Dato: " . date('d/m/Y H:i') . "</p>";
?>
```

Siden er straks tilgængelig på: `https://192.168.0.200/minside.php`

Genstart Nginx

```
sudo systemctl restart nginx
# eller blot reload (uden nedetid):
sudo systemctl reload nginx
```

Se Nginx logfiler

```
# Adgangslog (hvem har besøgt siden):
sudo tail -f /var/log/nginx/access.log

# Fejllog:
sudo tail -f /var/log/nginx/error.log
```

Konfigurationsfil

Nginx konfigurationen ligger i:

`/etc/nginx/sites-available/default`

Efter ændringer skal konfigurationen testes og Nginx reloades:

```
sudo nginx -t && sudo systemctl reload nginx
```

PHP 8.4

Hvad er PHP?

PHP kører server-side scripts — dvs. kode der udføres på RPI'en, inden svaret sendes til browseren. Det bruges typisk til dynamiske hjemmesider, formularer og databasekald.

Test at PHP virker

```
php -r "echo PHP_VERSION . PHP_EOL;"
```

Installerede PHP-moduler

Følgende moduler er installeret og aktive:

- **curl** — hent data fra eksterne URL'er
- **mbstring** — multibyte tekst (UTF-8)
- **mysql** — forbind til MariaDB
- **xml** — XML-parsing
- **zip** — zip-arkiver
- **opcache** — bytecode cache (hurtigere PHP)

Tilføj flere moduler med:

```
sudo apt install php8.4-[modulnavn]
sudo systemctl restart php8.4-fpm
```

Forbind PHP til MariaDB

```
<?php
$pdo = new PDO(
    'mysql:unix_socket=/run/mysqld/mysqld.sock;dbname=min_database',
    'brugernavn',
    'password'
);
$stmt = $pdo->query("SELECT * FROM min_tabel");
while ($row = $stmt->fetch()) {
    echo $row['felt'] . "<br>";
}
?>
```

PHP konfiguration

`/etc/php/8.4/fpm/php.ini`

Efter ændringer i php.ini:

```
sudo systemctl restart php8.4-fpm
```

MariaDB — Database

Hvad er MariaDB?

MariaDB er en relationsdatabase (MySQL-kompatibel) der bruges til at gemme og hente struktureret data — fx brugerdata, sensorværdier, konfigurationer.

Log ind via terminal

```
sudo mariadb
```

Du er nu i MariaDB-prompten som root.

Grundlæggende kommandoer

```
-- Se alle databaser
SHOW DATABASES;

-- Opret en ny database
CREATE DATABASE mit_projekt;

-- Vælg database
USE mit_projekt;

-- Opret en tabel
CREATE TABLE sensorer (
    id INT AUTO_INCREMENT PRIMARY KEY,
    navn VARCHAR(100),
    vaerdi FLOAT,
    tidspunkt DATETIME DEFAULT NOW()
);

-- Indsæt data
INSERT INTO sensorer (navn, vaerdi) VALUES ('temperatur', 22.5);

-- Hent data
SELECT * FROM sensorer;

-- Afslut
EXIT;
```

Opret ny databasebruger

```
-- Log ind som root
sudo mariadb

-- Opret bruger med password
CREATE USER 'minbruger'@'localhost' IDENTIFIED BY 'mitpassword';

-- Giv adgang til specifik database
GRANT ALL PRIVILEGES ON mit_projekt.* TO 'minbruger'@'localhost';

FLUSH PRIVILEGES;
```

Backup og restore

```
# Backup en database til fil
mysqldump -u root mit_projekt > backup.sql

# Restore fra fil
sudo mariadb mit_projekt < backup.sql
```

phpMyAdmin — Database via Browser

Hvad er phpMyAdmin?

phpMyAdmin er et web-baseret grafisk interface til MariaDB. Du kan oprette databaser, tabeller, køre SQL-forespørgsler og administrere brugere — alt via browseren.

Adgang

`https://192.168.0.200/phpmyadmin/`

Første login — opret en admin-bruger

Da root-kontoen bruger socket-authentication, skal du oprette en bruger til phpMyAdmin:

```
sudo mariadb
```

```
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'dit-password';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;
EXIT;
```

Log derefter ind i phpMyAdmin med `admin` og dit valgte password.

Hvad kan du gøre i phpMyAdmin?

- **Oprette** databaser og tabeller med klik
- **Importere** SQL-filer (fx backup)
- **Eksportere** databaser til SQL, CSV, Excel
- **Køre SQL-forespørgsler** direkte
- **Se og redigere** tabeldata
- **Administrere** brugere og rettigheder

Flask Dashboard — Python

Hvad er Flask?

Flask er et letvægts Python web-framework. Det bruges her til at bygge et dashboard der kan vise data fra MQTT, databaser og andre kilder.

App-placering

`/var/www/dashboard/app.py`

Redigér dashboard-appen

```
nano /var/www/dashboard/app.py
```

Grundlæggende Flask-struktur

```
from flask import Flask, render_template_string

app = Flask(__name__)

@app.route('/')
def index():
    return "<h1>Mit Dashboard</h1>"

@app.route('/data')
def data():
    return {"temperatur": 22.5, "luftfugtighed": 60}
```

Genstart efter ændringer

```
sudo systemctl restart flask-dashboard
```

Se Flask logfiler

```
sudo journalctl -u flask-dashboard -f
```

Status og fejlfinding

```
# Tjek om Flask kører
sudo systemctl status flask-dashboard

# Test direkte (uden Nginx)
curl http://127.0.0.1:5000/
```

Tilføj Python-pakker

```
sudo apt install python3-[pakkenavn]
# eller via pip i et virtuelt miljø:
python3 -m venv /var/www/dashboard/venv
source /var/www/dashboard/venv/bin/activate
pip install flask-mqtt sqlalchemy
```

Hvis du bruger et virtuelt miljø, skal du opdatere systemd-servicen:

```
sudo nano /etc/systemd/system/flask-dashboard.service
# Skift ExecStart til:
# ExecStart=/var/www/dashboard/venv/bin/gunicorn ...
sudo systemctl daemon-reload && sudo systemctl restart flask-dashboard
```

Mosquitto — MQTT Broker

Hvad er MQTT?

MQTT er en letvægts kommunikationsprotokol til IoT-enheder (sensorer, aktuatorer, Arduino, ESP32 osv.). Enheder **publicerer** data til et emne (topic), og andre enheder **abonnerer** på det emne og modtager dataene.

Forbindelsesoplysninger

Indstilling	Værdi
Server (broker)	192.168.0.200
Port (MQTT)	1883
Port (WebSockets)	9001
Brugernavn	mqtt
Password	Tommy2979#

Topics — navnekonvention

Topics er hierarkiske stier adskilt med `/`:

```
hjem/stue/temperatur
hjem/stue/luftfugtighed
hjem/garage/lys
sensor/esp32-01/data
```

Wildcard `#` matcher alt nedenunder:

```
hjem/#        → alle topics under hjem/
hjem/stue/+   → alle topics ét niveau under hjem/stue/
```

Test fra terminal (på RPI)

```
# Publicér en besked
mosquitto_pub -h localhost -u mqtt -P 'Tommy2979#' \
  -t hjem/stue/temperatur -m "22.5"

# Abonnér på et topic (lyt efter beskeder)
mosquitto_sub -h localhost -u mqtt -P 'Tommy2979#' \
  -t hjem/stue/temperatur

# Abonnér på ALT
mosquitto_sub -h localhost -u mqtt -P 'Tommy2979#' -t '#'
```

Test fra en anden PC/enhed på netværket

```
# Publicér
mosquitto_pub -h 192.168.0.200 -u mqtt -P 'Tommy2979#' \
  -t test/ping -m "hej"

# Abonnér
mosquitto_sub -h 192.168.0.200 -u mqtt -P 'Tommy2979#' \
  -t test/#
```

Tilføj en ny MQTT-bruger

```
# Tilføj bruger (tilføj til eksisterende fil)
sudo mosquitto_passwd /etc/mosquitto/passwd nybruger

# Genstart Mosquitto
sudo systemctl restart mosquitto
```

MQTT fra Python (Flask dashboard)

Installer paho-mqtt:

```
sudo apt install python3-paho-mqtt
```

```
import paho.mqtt.client as mqtt

def on_message(client, userdata, msg):
    print(f"{msg.topic}: {msg.payload.decode()}")

client = mqtt.Client()
client.username_pw_set("mqtt", "Tommy2979#")
client.on_message = on_message
client.connect("127.0.0.1", 1883)
client.subscribe("hjem/#")
client.loop_forever()
```

MQTT fra ESP32/Arduino

```
#include <PubSubClient.h>

const char* mqtt_server = "192.168.0.200";
const char* mqtt_user   = "mqtt";
const char* mqtt_pass   = "Tommy2979#";

client.connect("ESP32", mqtt_user, mqtt_pass);
client.publish("hjem/sensor/temp", "22.5");
```

Se Mosquitto logfil

```
sudo tail -f /var/log/mosquitto/mosquitto.log
```

UFW — Firewall

Hvad er UFW?

UFW (Uncomplicated Firewall) er en firewall der styrer hvilken netværkstrafik der tillades ind og ud af RPI'en. Alle porte er blokeret som standard — kun de specificerede er åbne.

Aktive regler

```
sudo ufw status verbose
```

Port	Tilladt	Formål
22	Ja	SSH fjernadgang
80	Ja	HTTP (redirect til HTTPS)
443	Ja	HTTPS webserver
1883	Ja	MQTT
9001	Ja	MQTT WebSockets

Tilføj en ny regel

```
# Tillad en port
sudo ufw allow 8080/tcp comment 'Min app'

# Tillad kun fra specifik IP
sudo ufw allow from 192.168.0.10 to any port 3306 comment 'MariaDB fra min PC'
```

Fjern en regel

```
# Se nummererede regler
sudo ufw status numbered

# Slet regel nummer 6
sudo ufw delete 6
```

Aktivér/deaktivér firewall

```
sudo ufw enable   # tænd
sudo ufw disable  # sluk (ikke anbefalet)
```

Git — Versionsstyring

Hvad er Git?

Git holder styr på ændringer i dine filer. Du kan se historik, fortryde ændringer og samarbejde med andre.

Opsæt Git (første gang)

```
git config --global user.name "Dit Navn"
git config --global user.email "din@email.dk"
```

Opret et nyt repository

```
cd /var/www/dashboard
git init
git add .
git commit -m "Første version af dashboard"
```

Vigtigste Git-kommandoer

```
# Se status på ændrede filer
git status

# Tilføj filer til næste commit
git add app.py
git add .           # tilføj alle filer

# Gem ændringer
git commit -m "Tilføjet temperatur-widget"

# Se historik
git log --oneline

# Fortryd ændringer i en fil (tilbage til seneste commit)
git checkout -- app.py

# Se hvad der er ændret
git diff
```

SSL-certifikat

Oversigt

Certifikatet er udstedt af en lokal CA (Certificate Authority) oprettet med **mkcert**. Browsere stoler på certifikatet, når CA-certifikatet er installeret på klientenheden.

Certifikatoplysninger

CA-rod	`/home/web/.local/share/mkcert/rootCA.pem`
Server-certifikat	`/etc/nginx/ssl/selfsigned.crt`
Server-nøgle	`/etc/nginx/ssl/selfsigned.key`
Udløber	28. juni 2028
Gyldigt for	192.168.0.200, raspberrypi, raspberrypi.local, localhost

Tilføj CA-certifikat på ny Windows-PC

1. Gå til `https://192.168.0.200/rpi-ca.crt` i Chrome
2. Klik "Avanceret → Fortsæt" for at komme ind
3. Filen downloades automatisk
4. Dobbeltklik på `rpi-ca.crt`
5. Installér certifikat → Lokal computer
6. Vælg "Betroede rodcertificeringsinstanser"
7. Genstart Chrome: `chrome://restart`

Forny certifikatet (inden udløb i 2028)

```
cd /tmp
mkcert 192.168.0.200 raspberrypi raspberrypi.local localhost 127.0.0.1
sudo cp 192.168.0.200+4.pem /etc/nginx/ssl/selfsigned.crt
sudo cp 192.168.0.200+4-key.pem /etc/nginx/ssl/selfsigned.key
sudo systemctl reload nginx
```

Daglig drift

Start/stop af tjenester

```
sudo systemctl start|stop|restart|status nginx
sudo systemctl start|stop|restart|status php8.4-fpm
sudo systemctl start|stop|restart|status mariadb
sudo systemctl start|stop|restart|status mosquitto
sudo systemctl start|stop|restart|status flask-dashboard
```

Tjek systemressourcer

```
# CPU, RAM, load
htop

# Diskplads
df -h

# RAM
free -h

# Kørende processer
ps aux | grep nginx
```

Vigtige logfiler

```
# Nginx adgangslog
sudo tail -f /var/log/nginx/access.log

# Nginx fejllog
sudo tail -f /var/log/nginx/error.log

# Mosquitto log
sudo tail -f /var/log/mosquitto/mosquitto.log

# Flask log
sudo journalctl -u flask-dashboard -f

# Systemlog
sudo journalctl -f
```

Opdatér systemet

```
sudo apt update && sudo apt upgrade -y
```

Reboot

```
sudo reboot
```

Alle tjenester starter automatisk igen efter reboot.