Prüfungsvorbereitung „Übung PI“

Dieses Dokument zeigt eine Mitschrift zur Ausarbeitung eines Arbeitsauftrage/Laborauftrages im Unterrichtsfach **Betriebssysteme & Labor** (Hr. Ullner)

Inhalt

1. Zielsetzung

Prüfungsbeispiele

2. Geräteübersicht

	Device (# & Name)	Rolle	Hypervisor (VM)	Betriebssystem (OS)	IP-Adresse (LAN)
1	Raspberry PI	Server	nein	Ubuntu 24.10	192.168.1.15
2					
3					

2.1 Netzwerkeinstellungen (Hardware)

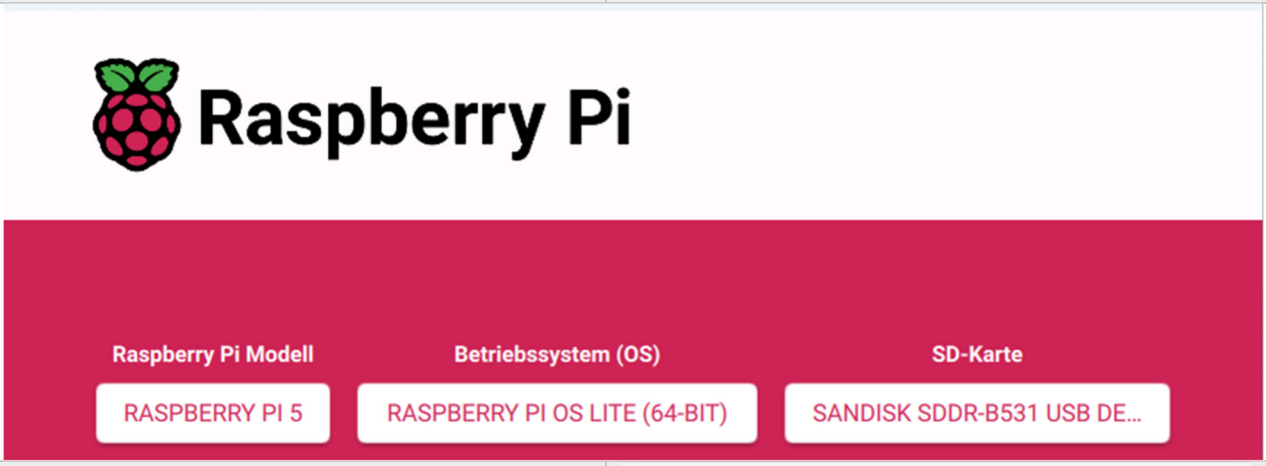
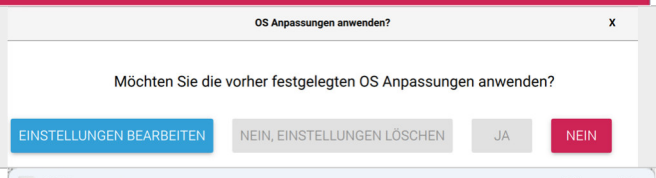
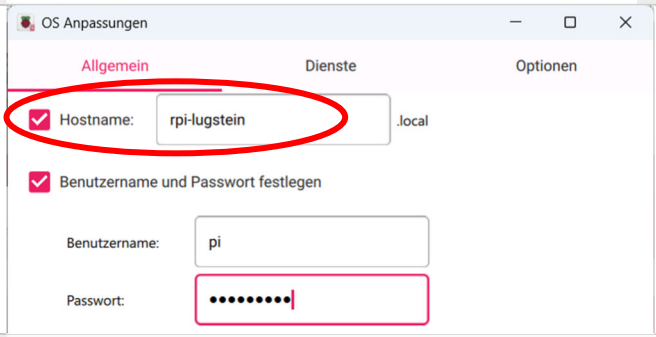
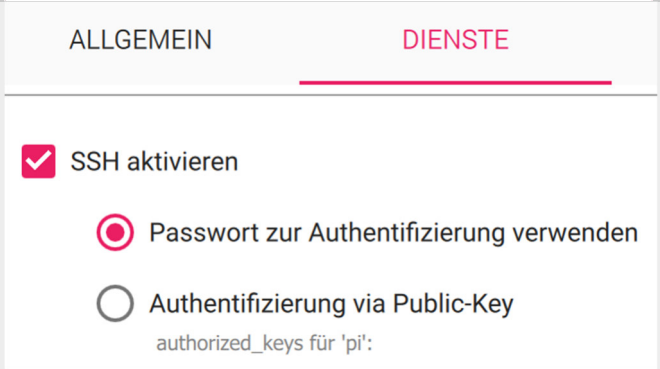
Devices	Device #	# 1	# 2	# 3	# 4	# 5
	IP Adr.	192.168.1.15				
	DHCP/static	DHCP				
	SUB	255.255.255.0				
	Gateway	192.168.1.1				
	DNS					
	Mac					

2.2 User & Zugangsdaten (Software)

Client	Geräte #	Anwendung	Benutzername	Passwort	Gerätename	domain (AD)
	PI	OS	pi	Aa_123456		
	Webmin		pi	Aa_123456		

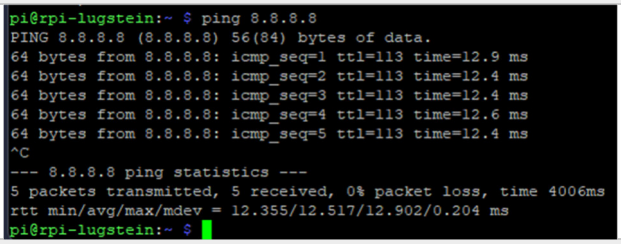
Server	Geräte #	Anwendung	Benutzername	Passwort	Hostname

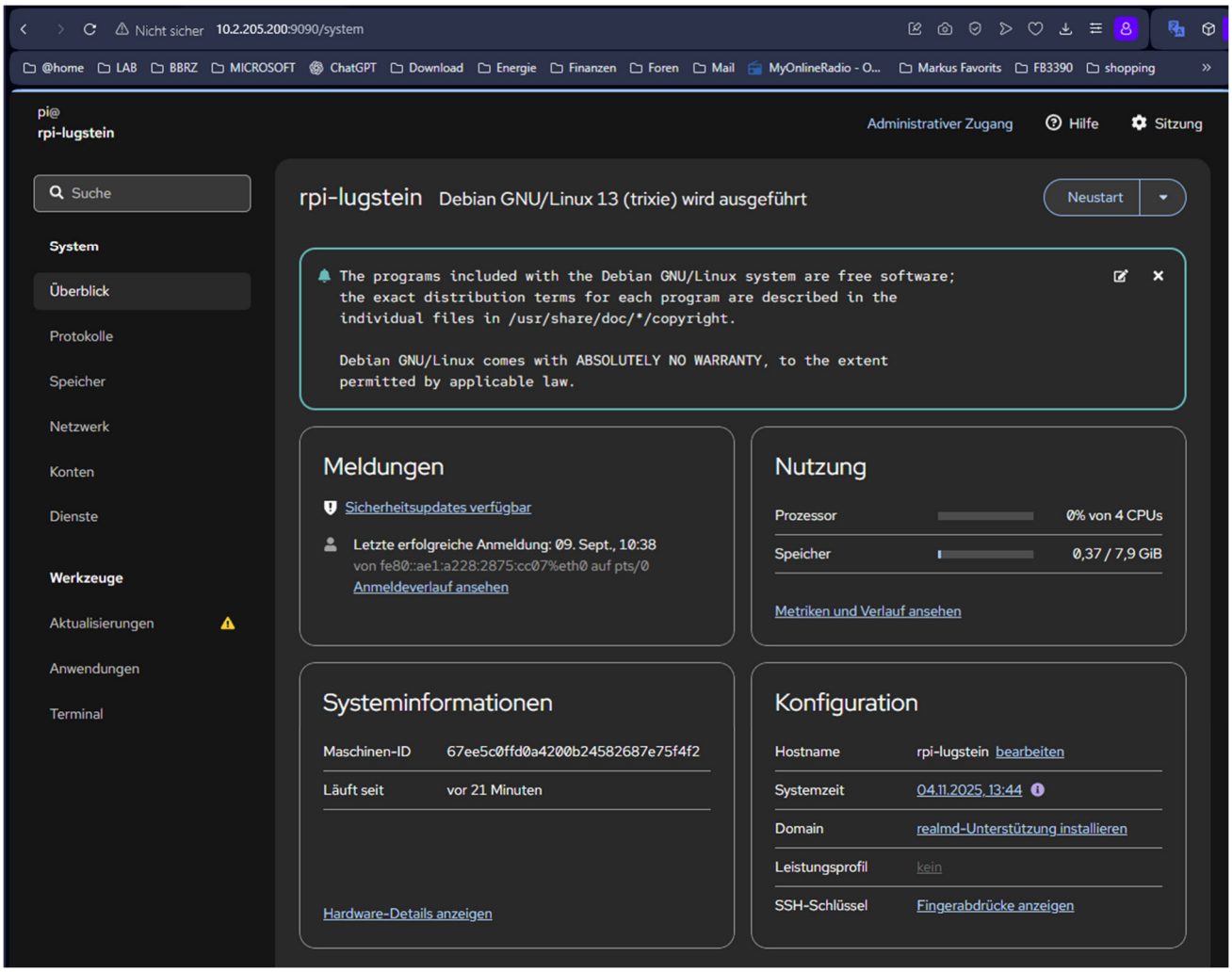
3. Installation

Erklärung	Beispiel
Auswahl eines geeigneten Betriebssystems	
	
Einstellungen bearbeiten	
Unter **„Allgemein“** Eingabe des Hostnames pilumex5 Festlegen von Benutzernamen & Passwort PW: Aa_123456 Hostname bereits hier vergeben!!!	
Unter **„Dienste“** Aktivierung von SSH	
Bestätigung des Schreibvorganges mit JA	

Wurde der Schreibvorgang beendet, kann die SD-Karte in den Raspi gesteckt und das System gestartet werden.	**Raspberry Pi OS (Legacy, 64-bit) Lite** wurde auf **SanDisk SDDR-B531 USB Device** geschrieben Sie können die SD-Karte nun aus dem Lesegerät entfernen WEITER

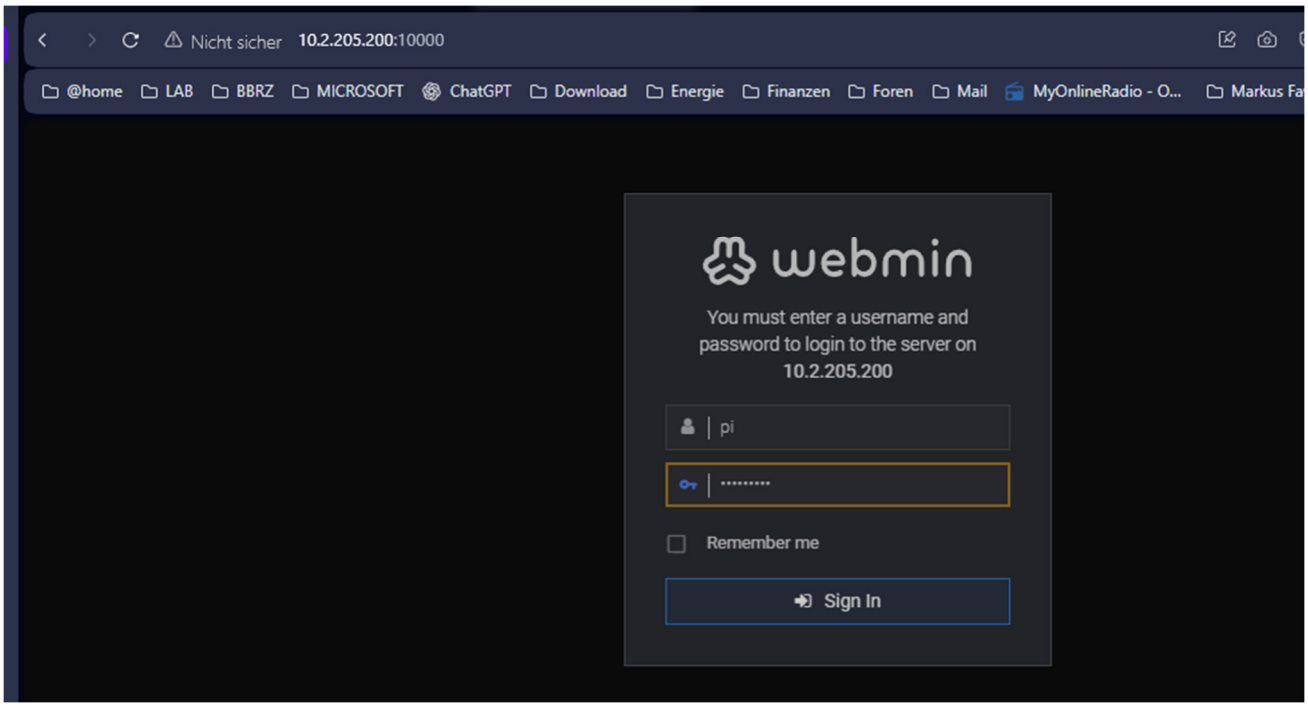
3.1 IP- Adresse statisch setzen

sudo apt-get update && sudo apt-get upgrade	System aktualisieren
sudo nmcli connection show	zeigt sämtliche Netzwerkschnittstellen an
ip addr	abrufen der IP-Adresse
sudo nmcli connection modify "Wired connection 1" ipv4.method manual ipv4.addresses 10.2.205.200/24 ipv4.gateway 10.2.205.254 ipv4.dns 8.8.8.8,1.1.1.1	
sudo nmcli con down Wired\ connection\ 1 && sudo nmcli con up Wired\ connection\ 1	
Cockpit installieren	sudo apt-install cockpit
Anmeldung: pi **PW: Aa_123456**	



Webmin installieren

Beschreibung	Syntax
das offizielle Webmin-Setup-Skript herunterladen, das das Repository einrichtet und die notwendigen GPG-Schlüssel installiert	curl -o webmin-setup-repos.sh https://raw.githubusercontent.com/webmin/webmin/master/webmin-setup-repos.sh
Skript ausführen und bestätigen der Einrichtung des Repositories mit y, wenn man dazu aufgefordert wird.	sudo sh webmin-setup-repos.sh
Webmin installieren	sudo apt install webmin --install-recommends
System aktualisieren	sudo apt update && sudo apt upgrade
Login „pi“; PW: Aa_123456	https://10.2.205.200:10000



Samba installieren

Installieren der Samba-Pakete	sudo apt-get install samba samba-common-bin

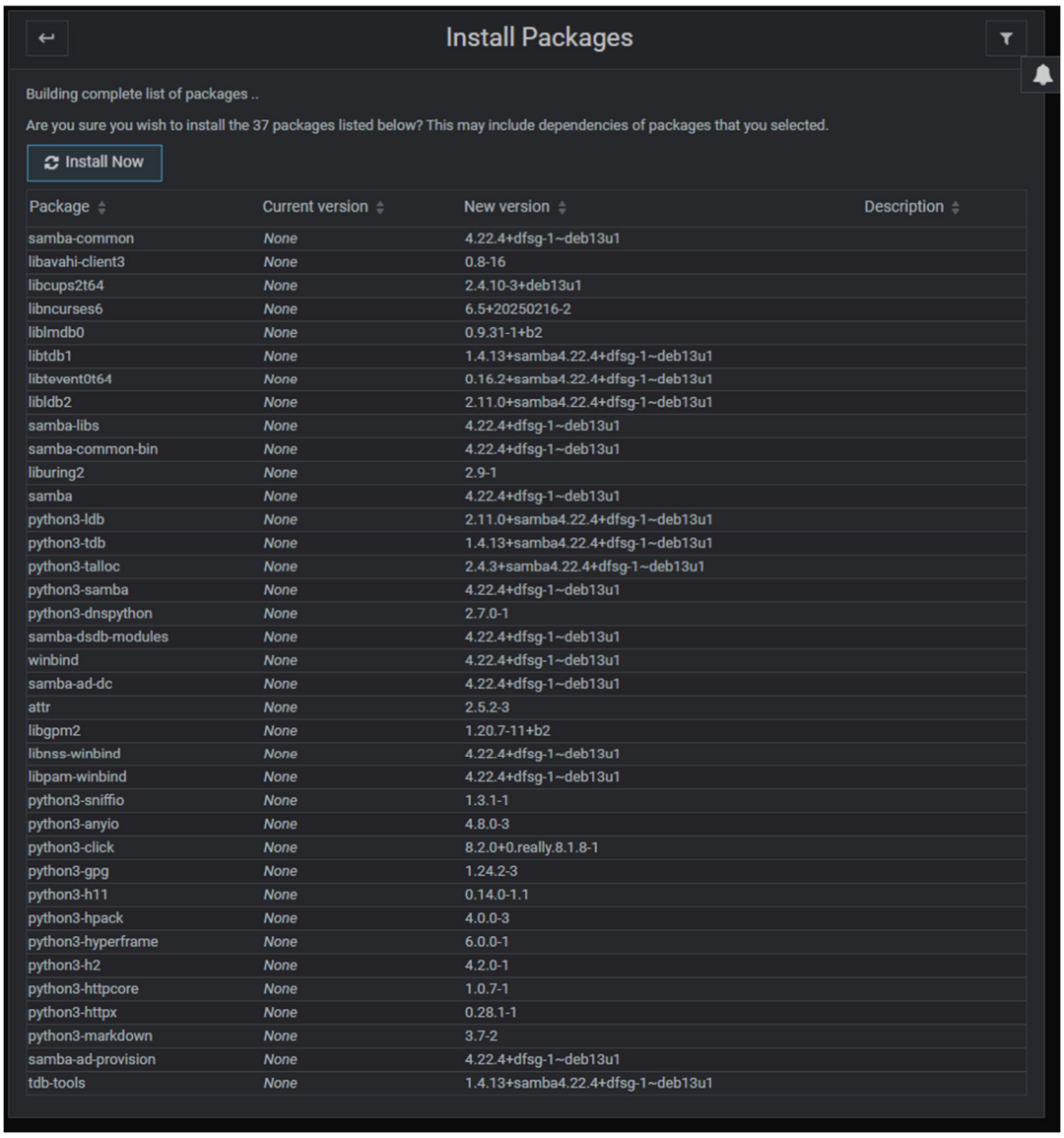
Benutzer und Gruppen über Webmin erstellt

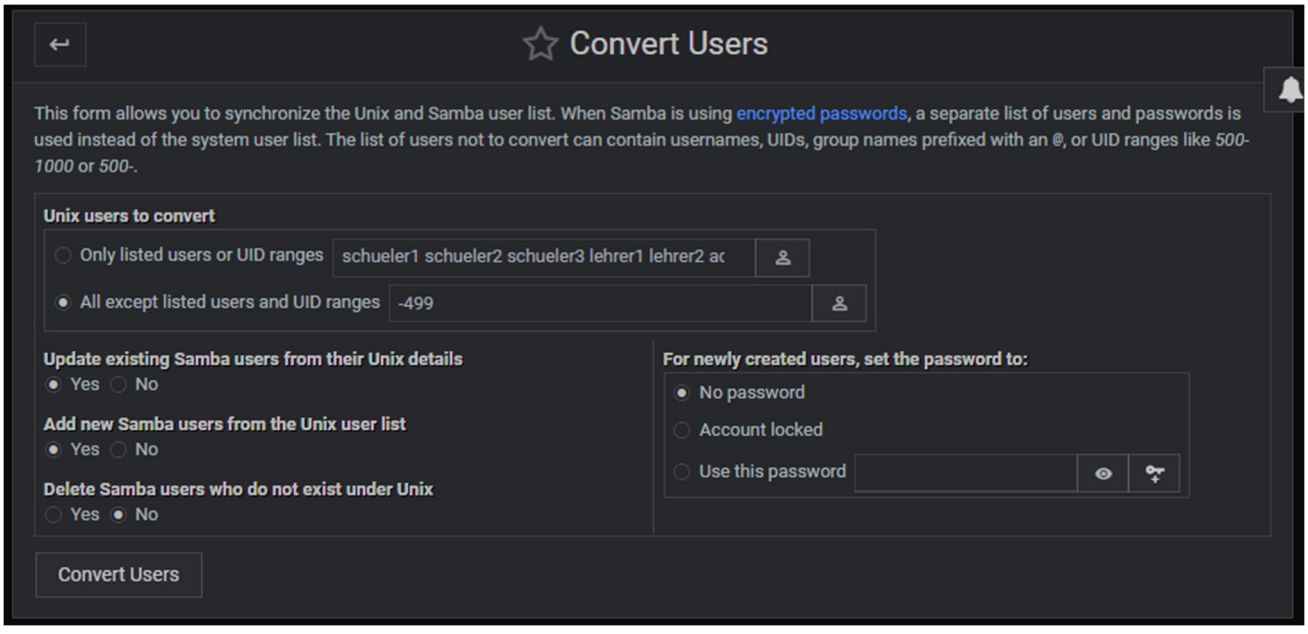
```
pi@rpi-lugstein:~ $ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
dhcpcd:x:100:65534:DHCP Client Daemon:/usr/lib/dhcpcd:/bin/false
pi:x:1000:1000::/home/pi:/bin/bash
systemd-timesync:x:991:991:systemd Time Synchronization:/:/usr/sbin/nologin
messagebus:x:990:990:System Message Bus:/nonexistent:/usr/sbin/nologin
sshd:x:989:65534:sshd user:/run/sshd:/usr/sbin/nologin
avahi:x:101:103:Avahi mDNS daemon:/run/avahi-daemon:/usr/sbin/nologin
polkitd:x:988:988:User for polkitd:/:/usr/sbin/nologin
dnsmasq:x:999:65534:dnsmasq:/var/lib/misc:/usr/sbin/nologin
schueler1:x:1001:100::/home/schueler1:/bin/sh
schueler2:x:1002:100::/home/schueler2:/bin/sh
schueler3:x:1003:100::/home/schueler3:/bin/sh
lehrer1:x:1004:100::/home/lehrer1:/bin/sh
lehrer2:x:1005:100::/home/lehrer2:/bin/sh
admin:x:1006:100::/home/admin:/bin/sh
pi@rpi-lugstein:~ $
```

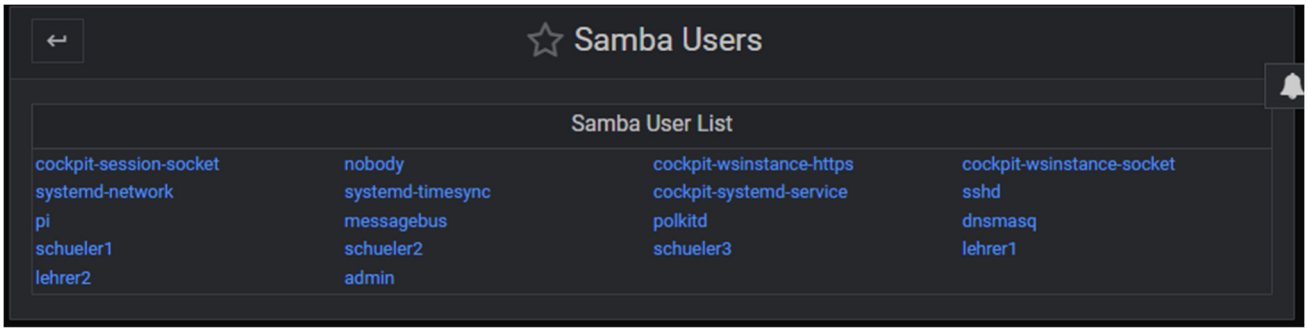
```
pi@rpi-lugstein:~ $ cat /etc/group
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:pi
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:pi
fax:x:21:
voice:x:22:
cdrom:x:24:pi
floppy:x:25:
tape:x:26:
sudo:x:27:pi
audio:x:29:pi
dip:x:30:
www-data:x:33:
backup:x:34:
operator:x:37:
list:x:38:
irc:x:39:
src:x:40:
shadow:x:42:
utmp:x:43:
video:x:44:pi
sasl:x:45:
plugdev:x:46:pi
staff:x:50:
games:x:60:pi
users:x:100:pi
nogroup:x:65534:
systemd-journal:x:999:
systemd-network:x:998:
crontab:x:997:
input:x:996:pi
sgx:x:995:
clock:x:994:
kvm:x:993:
render:x:992:pi
pi:x:1000:
systemd-timesync:x:991:
messagebus:x:990:
_ssh:x:101:
bluetooth:x:102:
avahi:x:103:
netdev:x:104:pi
polkitd:x:988:
spi:x:989:pi
i2c:x:987:pi
gpio:x:986:pi
schulgruppe:x:1001:schueler1,schueler2,schueler3
lehrergruppe:x:1002:lehrer1,lehrer2
projektgruppe:x:1003:schueler1,lehrer1,admin
admingruppe:x:1004:lehrer1,admin
pi@rpi-lugstein:~ $
```

Gruppenzugehörigkeiten	
	`pi@rpi-lugstein:~ $ groups schueler1` `schueler1 : users schulgruppe projektgruppe`
	`pi@rpi-lugstein:~ $ groups schueler2` `schueler2 : users schulgruppe`
	`pi@rpi-lugstein:~ $ groups schueler3` `schueler3 : users schulgruppe`
	`pi@rpi-lugstein:~ $ groups lehrer1` `lehrer1 : users lehrergruppe projektgruppe admingruppe`
	`pi@rpi-lugstein:~ $ groups lehrer2` `lehrer2 : users lehrergruppe`
	`pi@rpi-lugstein:~ $ groups admin` `admin : users projektgruppe admingruppe`

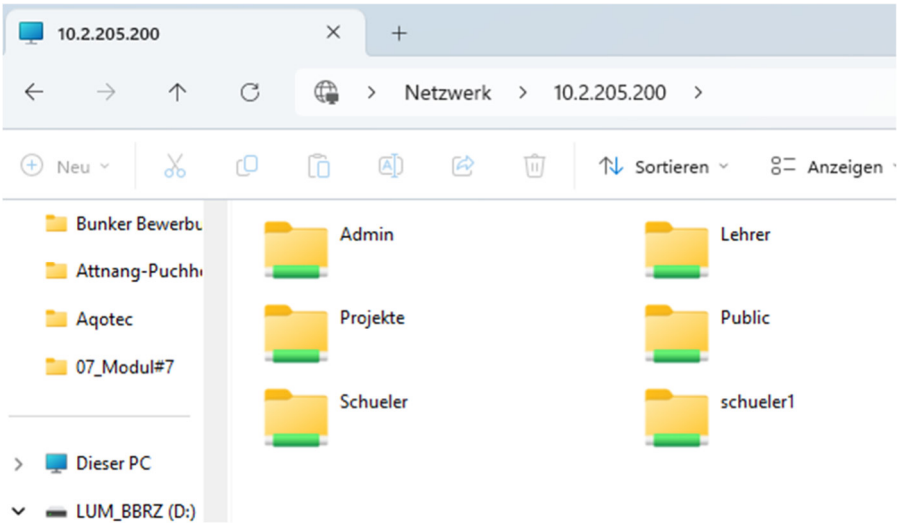
Installieren Samba per Webmin








```
;   write list = root, @lpadmin


[Public]
        comment = Austausch von nicht-sensiblen Daten
        path = /public
        public = yes

[Schueler]
        path = /schueler
        valid users = @schulgruppe
        read list = @schulgruppe
        write list = @schulgruppe

[Lehrer]
        path = /lehrer
        valid users = @lehrergruppe
        read list = @lehrergruppe
        write list = @lehrergruppe

[Projekte]
        path = /projekte
        valid users = @projektgruppe
        read list = @projektgruppe
        write list = @projektgruppe

[Admin]
        path = /admin
        valis usres = @admingruppe
        read list = @admingruppe
        write list = @admingruppe
```



am besten man erstellt die Samba-User im Terminal

sudo smbpasswd -a schueler1

```
pi@rpi-lugstein4:/ $ sudo smbpasswd  -a schueler1
New SMB password:
Retype new SMB password:
Added user schueler1.
pi@rpi-lugstein4:/ $ sudo smbpasswd  -a schueler2
New SMB password:
Retype new SMB password:
Added user schueler2.
pi@rpi-lugstein4:/ $ sudo smbpasswd  -a schueler3
New SMB password:
Retype new SMB password:
Added user schueler3.
pi@rpi-lugstein4:/ $ sudo smbpasswd  -a lehrer1
New SMB password:
Retype new SMB password:
Added user lehrer1.
pi@rpi-lugstein4:/ $ sudo smbpasswd  -a lehrer2
New SMB password:
Retype new SMB password:
Added user lehrer2.
pi@rpi-lugstein4:/ $ sudo smbpasswd  -a admin
New SMB password:
Retype new SMB password:
Added user admin.
pi@rpi-lugstein4:/ $
```

Einträge in die smb.conf für gruppenbasierte Freigaben

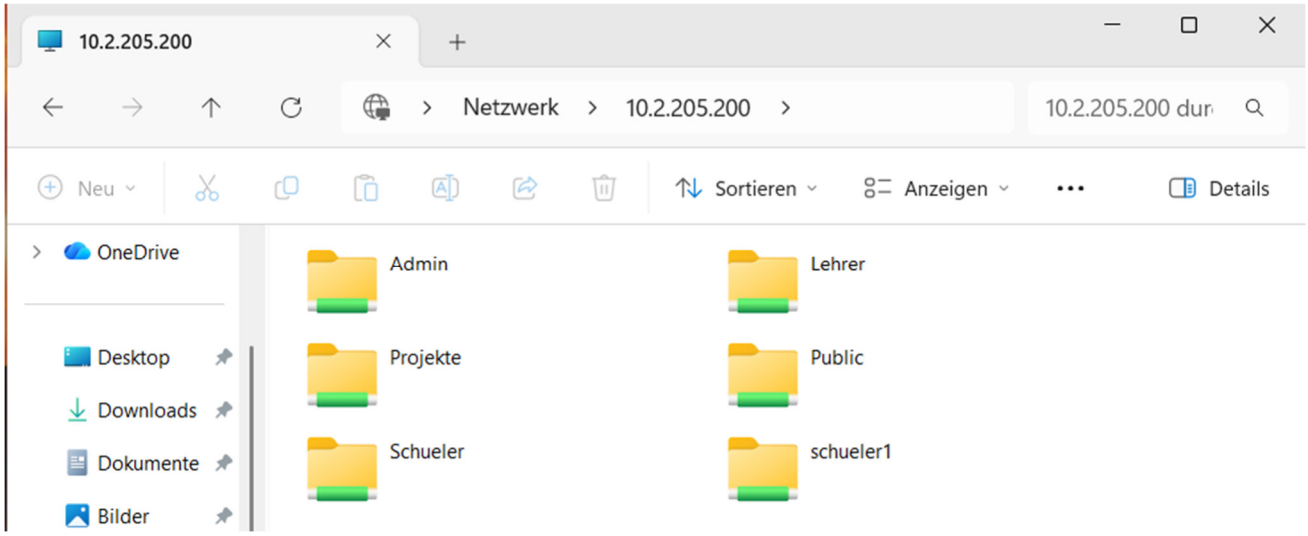
	[public] path = /public public = yes writable = yes browseable = yes guest ok = yes create mask = 0666 directory mask = 0777
Linux-Dateisystemrechte setzen	sudo mkdir -p /public sudo chmod -R 777 /public sudo chown -R nobody:nogroup /public
	[Schueler] path = /schueler browseable = yes writable = yes valid users = @schulgruppe create mask = 0660 directory mask = 2770

	force group = schulgruppe
Linux-Dateisystemrechte setzen	sudo mkdir -p /lehrer sudo chmod -R 777 /lehrer sudo chown -R nobody:nogroup /lehrer
	[Lehrer] path = /lehrer browseable = yes writable = yes valid users = @lehrergruppe create mask = 0660 directory mask = 2770 force group = lehrergruppe
Linux-Dateisystemrechte setzen	sudo mkdir -p /projekte sudo chmod -R 777 /projekte sudo chown -R nobody:nogroup /projekte
	[Projekte] path = /projekte browseable = yes writable = yes valid users = @projektgruppe create mask = 0660 directory mask = 2770 force group = projektgruppe
	sudo mkdir -p /admin sudo chown root:admingruppe /admin sudo chmod 2770 /admin
Linux-Dateisystemrechte setzen	[Admin] path = /admin browseable = yes writable = yes valid users = @admingruppe create mask = 0660 directory mask = 2770 force group = admingruppe
Samba-Dienst neu starten	sudo systemctl restart smbd
kombinierter Befehlsblock zur Rechte-Vergabe für alle Ordner	**# Ordner anlegen (falls sie noch nicht existieren)** sudo mkdir -p /public /schueler /lehrer /projekte /admin **# Eigentümer und Gruppen setzen** sudo chown root:schulgruppe /schueler && sudo chown root:lehrergruppe /lehrer && sudo chown root:projektgruppe /projekte && sudo chown root:admingruppe /admin

	# Rechte setzen (SGID-Bit, Vollzugriff für Gruppe) sudo chmod 2770 /schueler /lehrer /projekte /admin **# Öffentliche Freigabe mit Vollzugriff für alle** sudo chmod 777 /public sudo chown nobody:nogroup /public

Tests durchgeführt:

Testzugriff von einem WIN11-Client:



Testzugriff von einem Linux-Client:

```
lumex@lumexE16:~$ smbclient -L /10.2.205.200 -N

        Sharename       Type      Comment
        ---------       ----      -------
        print$          Disk      Printer Drivers
        Public          Disk
        Schueler        Disk
        Lehrer          Disk
        Projekte        Disk
        Admin           Disk
        IPC$            IPC       IPC Service (Samba 4.22.4-Debian-4.22.4+dfsg-1~deb13u1)
        nobody          Disk      Home Directories
SMB1 disabled -- no workgroup available
```

4. Apache Webserver mit Startseite

	sudo apt update
	sudo apt install apache2
Aufruf der Standard-Seite	http://10.2.205.200
Verzeichnis /var/www/html **neue Datei mit nano geöffnet**	index.html gesichert (mv-Befehl) sudo nano index.html
onlineeditor zum erstellen der persönlichen Startseite genutz.	
per Copy – Paste html Eintrag in der neuen index.html gemacht.	http://10.2.205.200

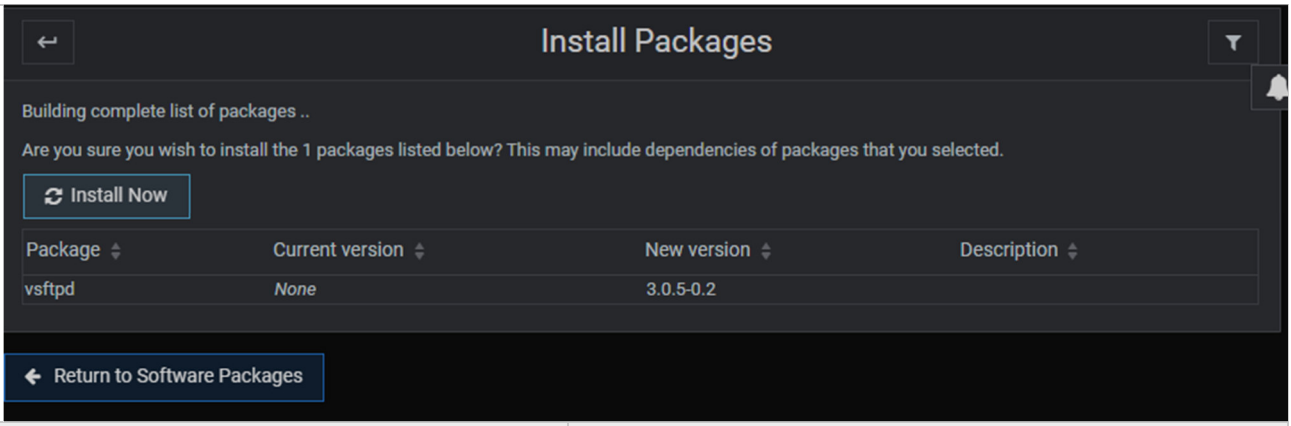
Ausgabe der Benutzer und Gruppen

```
admin:x:1006:100::/home/admin:/bin/sh
pi@rpi-lugstein4:/var/www/html $ awk -F: '$3 >= 1000 && $3 < 65534 { print $1 }' /etc/passwd
pi
schueler1
schueler2
schueler3
lehrer1
lehrer2
admin
pi@rpi-lugstein4:/var/www/html $ awk -F: '$3 >= 1000 && $3 < 65534 { print $1 }' /etc/group
pi
lehrergruppe
projektgruppe
schulgruppe
admingruppe
pi@rpi-lugstein4:/var/www/html $
```

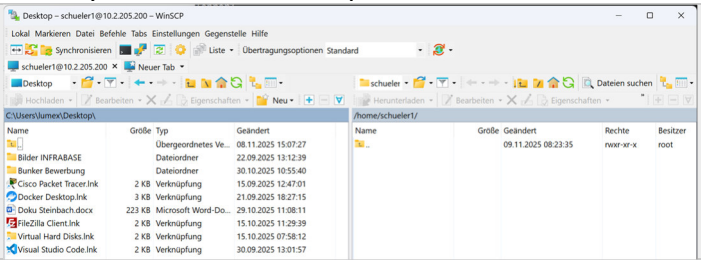
awk -F: '$3 >= 1000 && $3 < 65534 { print $1 }' /etc/passwd

5. FTP-Server

Installation per Webmin	System →




```
pi@rpi-lugstein4:/var/www/html $ sudo systemctl status vsftpd
● vsftpd.service - vsftpd FTP server
     Loaded: loaded (/usr/lib/systemd/system/vsftpd.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-11-10 08:19:38 GMT; 2min 9s ago
 Invocation: 2f4baa87486a4d9c96d4079d5ced6521
    Process: 7339 ExecStartPre=/bin/mkdir -p /var/run/vsftpd/empty (code=exited, status=0/SUCCESS)
   Main PID: 7341 (vsftpd)
      Tasks: 1 (limit: 3925)
        CPU: 54ms
     CGroup: /system.slice/vsftpd.service
             └─7341 /usr/sbin/vsftpd /etc/vsftpd.conf

Nov 10 08:19:38 rpi-lugstein4 systemd[1]: Starting vsftpd.service - vsftpd FTP server...
Nov 10 08:19:38 rpi-lugstein4 systemd[1]: Started vsftpd.service - vsftpd FTP server.
```

Konfiguration per Terminal öffnen der Datei /etc/vsftpd.conf	
Zugriff nur von lokalen Benutzern:	`# Uncomment this to allow local users to log in.` `local_enable=YES`
Deaktivierung von anonymen Anmeldungen:	`# Allow anonymous FTP? (Disabled by default).` `anonymous_enable=NO`
Zugriffs-Beschränkung bestimmter Benutzer auf ihr Home-Verzeichnisse	`# You may restrict local users to their home directories.  See the FAQ for` `# the possible risks in this before using chroot_local_user or` `# chroot_list_enable below.` `#chroot_local_user=YES`
enablen dieser Einstellungen:	`chroot_local_user=YES` `chroot_list_enable=YES` `# (default follows)` `chroot_list_file=/etc/vsftpd.chroot_list`
neue Datei mit dem Namen „vsftpd.chroot_list unter /etc erstellen.	pi@10.2.205.200:22 - `GNU nano 8.4` `schueler1` `schueler2` `schueler3` `lehrer1` `lehrer2` `admin`
Dienst neu starten	sudo systemctl restart vsftpd
Test per WINSCP	

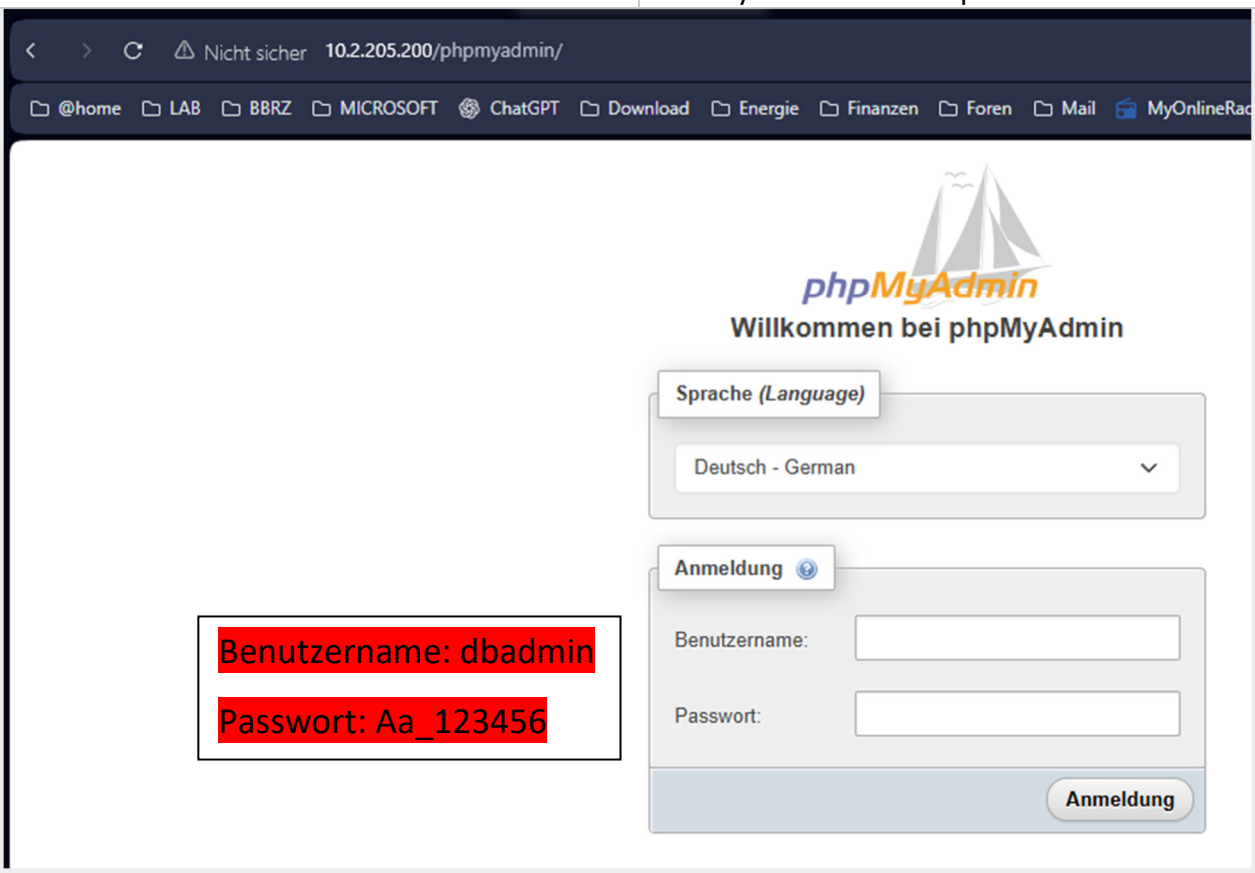
6. MySQL-Server

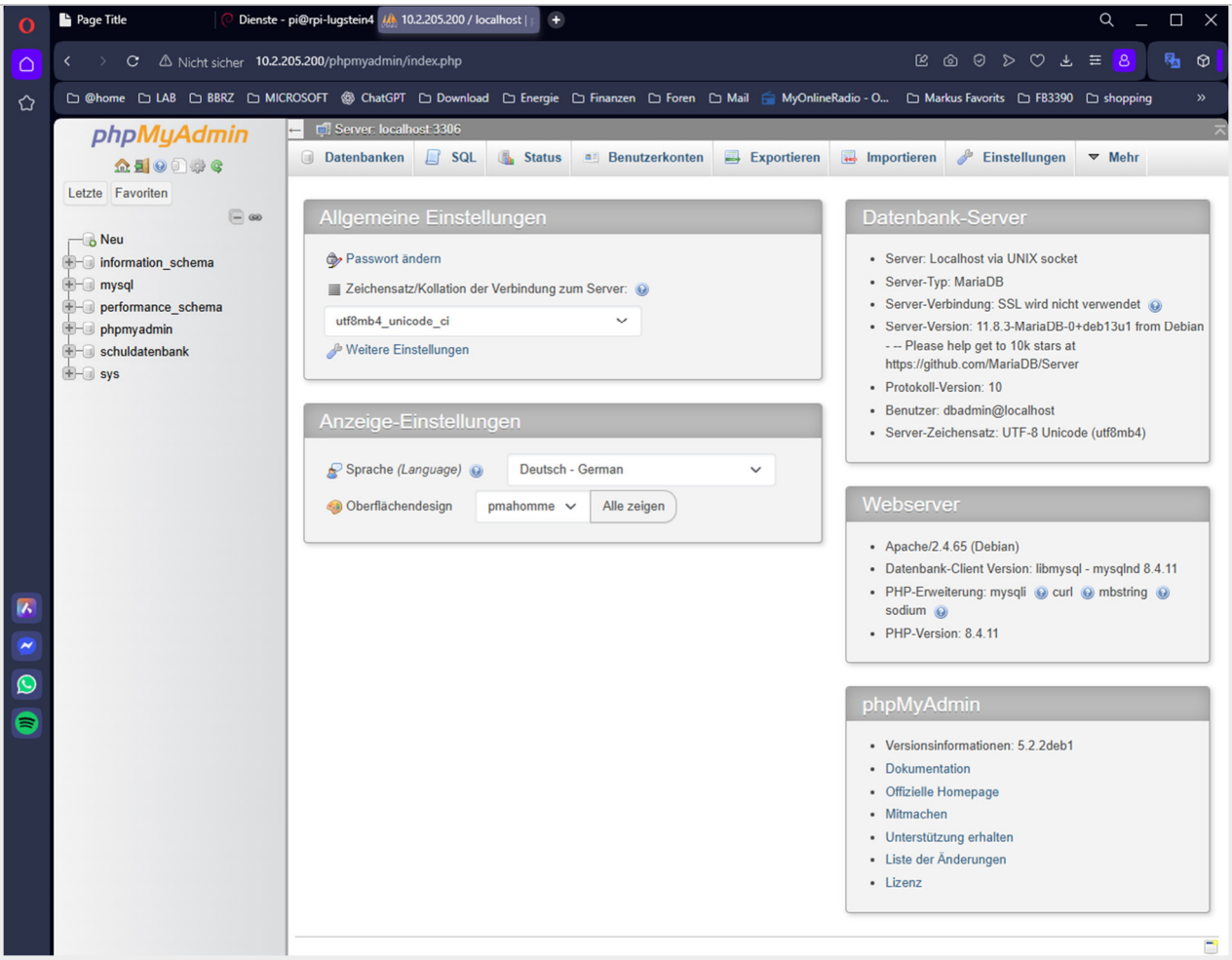
System-Aktualisierung	sudo apt update && sudo apt upgrade
installiert mariadb	sudo apt install mariadb-server
Sicherheitsconfiguration durchführen alles auf nein setzen	sudo mariadb-secure-installation
	pi@rpi-lugstein4:/ $ mariadb-secure-installation touch: cannot touch '.my.cnf.13842': Permission denied touch: cannot touch '.mysql.13842': Permission denied chmod: cannot access '.my.cnf.13842': No such file or directory chmod: cannot access '.mysql.13842': No such file or directory NOTE: MariaDB is secure by default in Debian. Running this script is useless at best, and misleading at worst. This script will be removed in a future MariaDB release in Debian. Please read mariadb-server.README.Debian for details. Enter root user password or leave blank: Enter current password for root (enter for none):
Login in die Datenbank:	sudo mysql -u root -p
	CREATE USER 'dbadmin'@'localhost' IDENTIFIED BY 'Aa_123456'; GRANT ALL PRIVILEGES ON *.* TO 'dbadmin'@'localhost' WITH GRANT OPTION;
	FLUSH PRIVILEGES;
	GRANT ALL PRIVILEGES ON *.* TO 'dbadmin'@'localhost';
	FLUSH PRIVILEGES;
erstellt die Datenbank	CREATE DATABASE schuldatenbank;
	USE schuldatenbank
	CREATE TABLE schueler_liste
zeigt die User	SELECT user FROM mysql.user;
dbadmin ist erstellt **schuldatenbank ist erstellt.** Tabelle schueler_liste existiert	MariaDB [(none)]> SELECT user FROM mysql.user; +-------------+ | User | +-------------+ | dbadmin | | mariadb.sys | | mysql | | root | +-------------+ 4 rows in set (0.002 sec) MariaDB [(none)]> CREATE DATABASE schuldatenbank; ERROR 1007 (HY000): Can't create database 'schuldatenbank'; database exists MariaDB [(none)]> SHOW DATABASES; +--------------------+ | Database | +--------------------+ | information_schema | | mysql | | performance_schema | | schuldatenbank | | sys | +--------------------+ 5 rows in set (0.001 sec) MariaDB [(none)]> HISTORY -> Exit -> ^C MariaDB [(none)]> exit Bye

```
MariaDB [schuldatenbank]> show tables;
+--------------------------+
| Tables_in_schuldatenbank |
+--------------------------+
| schueler_liste           |
+--------------------------+
1 row in set (0.001 sec)
```

7. phpMyAdmin

System aktualisieren	sudo apt update && sudo apt upgrade
php installieren	sudo apt install -y php libapache2-mod-php
php-Erweiterung installieren	sudo apt install -y php-mbstring php-zip php-gd php-mysql php-curl php-xml
Apache PHP-Modul aktivieren (falls nötig)	sudo a2enmod php8.2
Apache neu starten	sudo systemctl restart apache2
Konfiguration der Verbindung zu MySQL	
Voraussetzungen prüfen **➔ beide Dienste müssen laufen!**	sudo systemctl status mariadb sudo systemctl status apache2
phpMyAdmin in Apache aktivieren	sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available/phpmyadmin.conf sudo a2enconf phpmyadmin sudo systemctl reload apache2





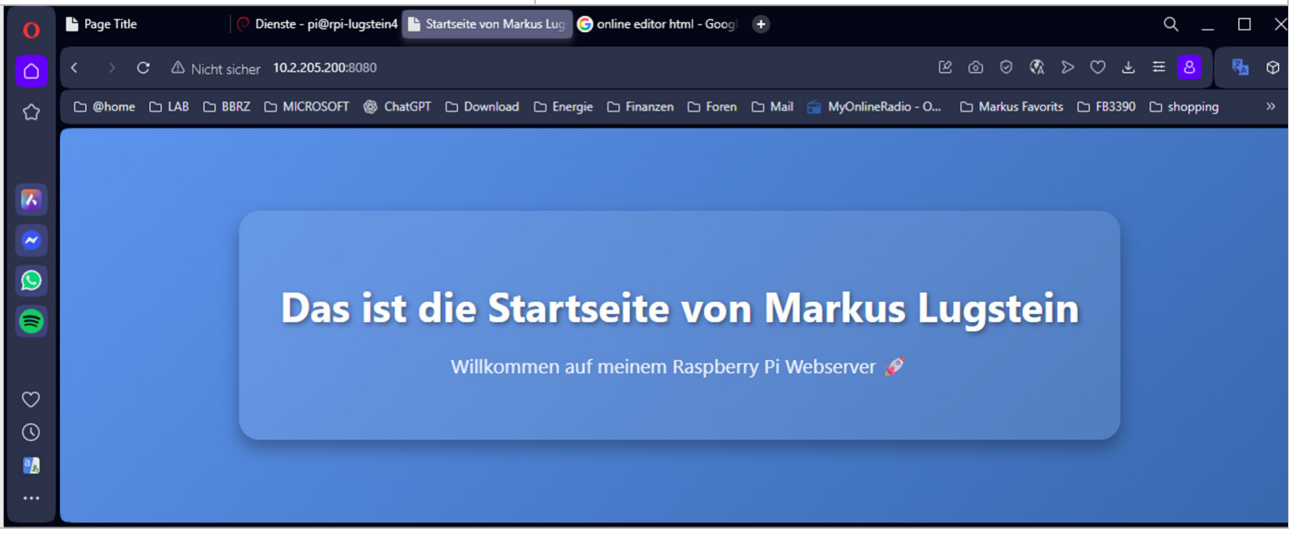
Zugriff per http://10.2.205.200/phpmyadmin

8. Docker Installation und Container-Management

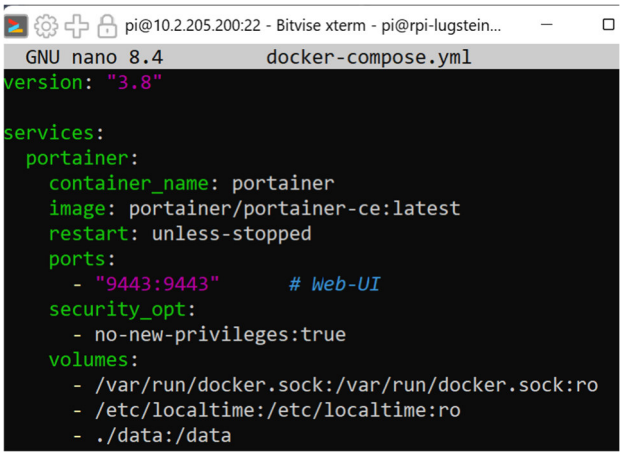
System aktualisieren	sudo apt-get update && sudo apt-get upgrade
installiert docker	curl -fsSL test.docker.com -o get-docker.sh && sh get-docker.sh
Nach der Installation prüfen ob docker läuft	sudo systemctl status docker
Benutzer zur Docker-Gruppe hinzufügen	sudo usermod -aG docker $USER
Docker Version	`pi@rpi-lugstein4:~ $ docker compose version` `Docker Compose version v2.40.3`

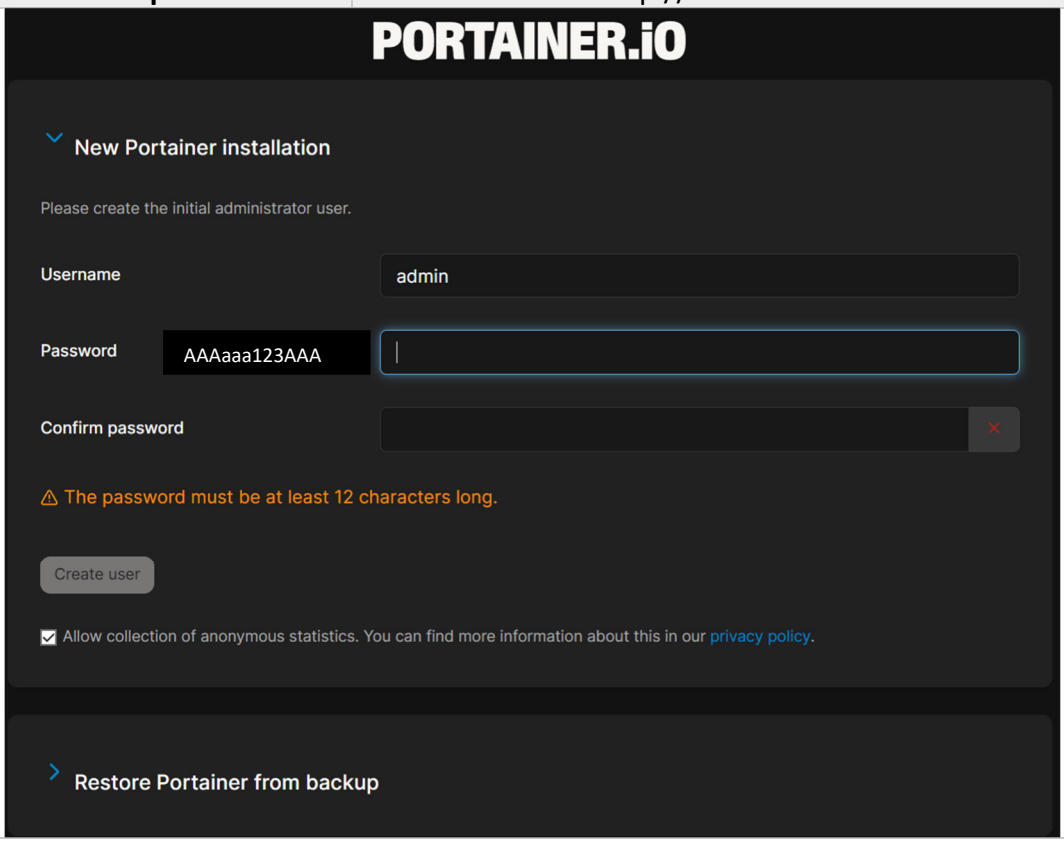
8.1 Container1: Nginx Webserver

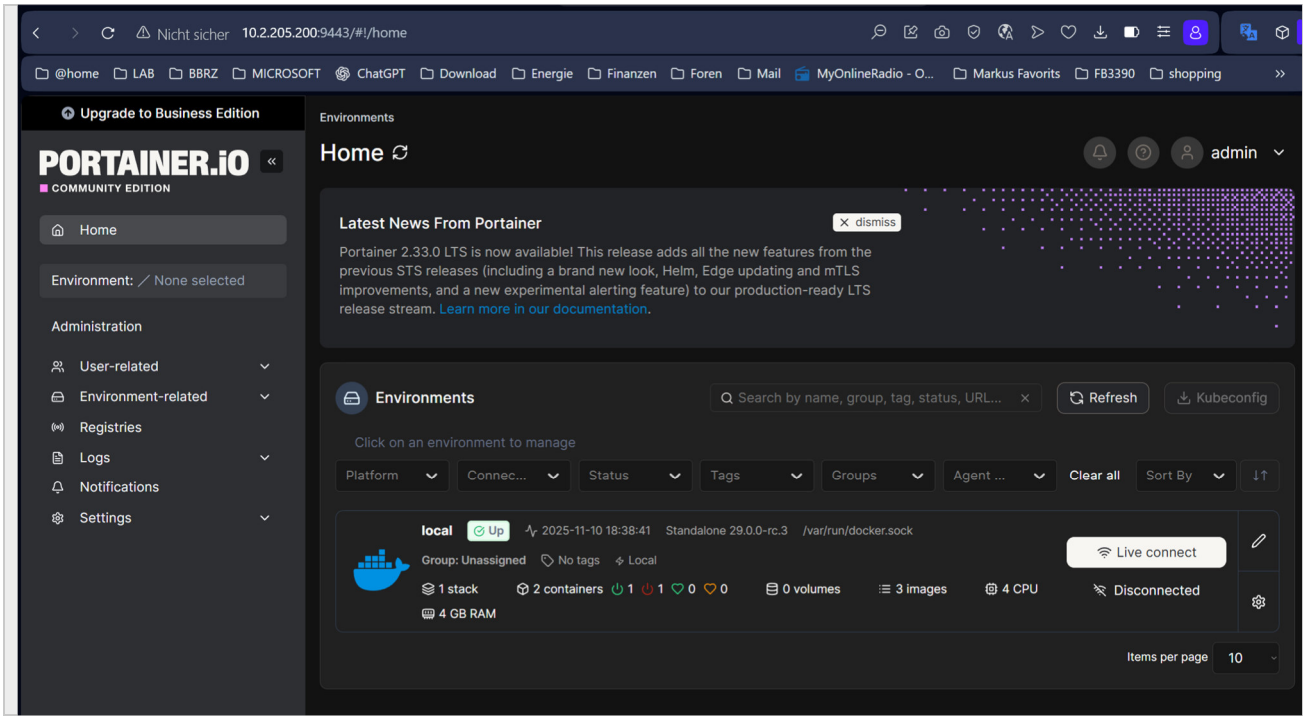
Dateistruktur erstellen → sudo mkdir -p /srv/docker/nginx	/srv/docker/nginx
Schreibrecht anpassen	sudo chown -R pi:pi /srv/docker
im gleichen Ordner „nginx“ mit nano die .yaml file erstellen	nano docker-compose.yml
Verzeichnis-Struktur	/srv/docker/nginx/
	/srv/docker/nginx/html/
Erstellung des .yml Files	version: '3.9' services: web: image: nginx:latest ports: - "8080:80"
Aufruf der Standar-Webseite ➔ http://10.2.205.200:8080	„Welcome to nginx!“ → Der Webserver funktioniert! ☑
Eigene Startseite erstellt	
Nginx-Container mit lokalem Webverzeichnis verbunden **Ergänzung des .yml files um den Eintrag „volumes“ Damit verwendet Nginx deneigenen HTML-Ordner.**	



8.2 Container2: Portainer (Docker Web-UI)

in das Verzeichnis wechseln	cd /srv/docker
	sudo mkdir portainer
wechselt in das Verzeichnis	cd /srv/docker/portainer
neuen Ordner „data“ erstellen	sudo mkdir data
erstellt neue Datei	sudo nano docker-compose.yml
Eintrag in die Datei: version: "3.8" services: portainer: container_name: portainer image: portainer/portainer-ce:latest restart: unless-stopped ports: - "9443:9443" # Web-UI security_opt: - no-new-privileges:true volumes: - /var/run/docker.sock:/var/run/docker.sock:ro - /etc/localtime:/etc/localtime:ro - ./data:/data	
in das Verzeichnis wechseln in der das .yml file liegt	cd /srv/docker/portainer
container starten	docker compose up -d
Aufruf im Browser per	http://10.2.205.200:9443





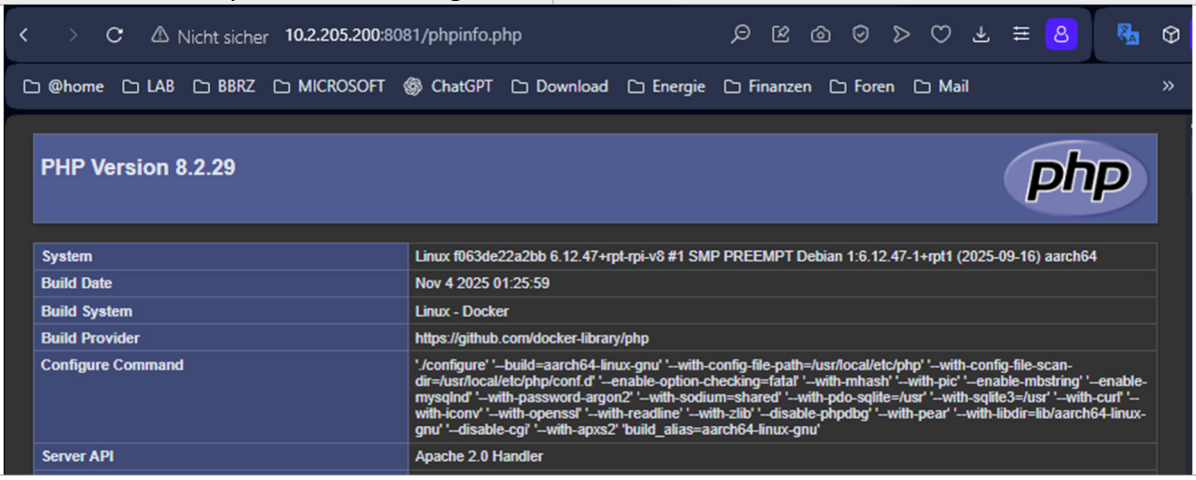
8.3 Container3: MySQL-Container

Erklärung	Beispiel
die Zeilen ports wurden auskommentiert da der port bereits von einer vorherigen Installation verwendet wird. **root-Passwort: Aa_123456** **Datenbank „labtest“ erstellt.**	pi@10.2.205.200:22 - Bitvise xterm -... GNU nano 8.4 docker-compose.yml version: "3.8" services: mysql: image: mariadb:10.11 container_name: mysql restart: unless-stopped environment: MYSQL_DATABASE: labtest MYSQL_ROOT_PASSWORD: Aa_123456 TZ: Europe/Vienna #ports: #- "3306:3306" # Nur aktiv lassen, volumes: - ./data/mysql:/var/lib/mysql - ./data/conf.d:/etc/mysql/conf.d - ./data/logs:/logs
container starten	docker compose up -d
Verbindung vom Host (Raspberry Pi) mit dem container hergestellt.	docker exec -it mysql mysql -uroot -p pi@rpi-lugstein4:/srv/docker/mysql $ docker exec -it mysql mysql -uroot -p Enter password: Welcome to the MariaDB monitor. Commands end with ; or \g. Your MariaDB connection id is 3 Server version: 10.11.14-MariaDB-ubu2204 mariadb.org binary distribution Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others. Type 'help;' or '\h' for help. Type '\c' to clear the current input statement. MariaDB [(none)]>
zusätzliche Datenbank erstellt. **CREATE DATABASE testdatenbank;**	MariaDB [(none)]> CREATE DATABASE testdatenbank; Query OK, 1 row affected (0.001 sec) MariaDB [(none)]>

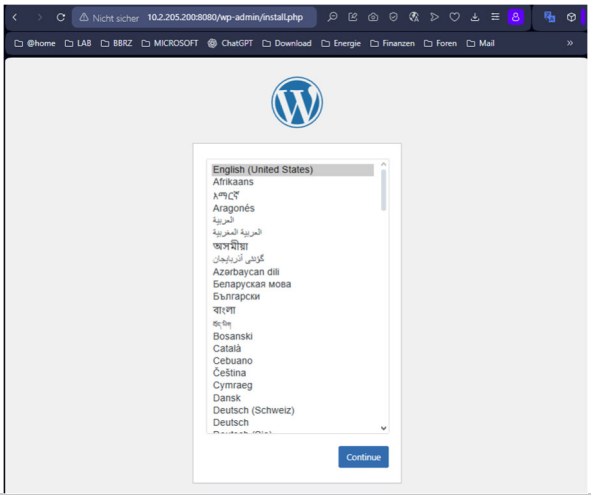
vorhandene Datenbanken	MariaDB [(none)]> SHOW DATABASES; +--------------------+ | Database | +--------------------+ | information_schema | | labtest | | mysql | | performance_schema | | sys | | testdatenbank | +--------------------+ 6 rows in set (0.001 sec)

8.4 Container4: Apache mit PHP (eigener Container)

Erklärung	Beispiel
Dateistruktur	pi@rpi-lugstein4:/srv/docker/apache_php $. ├── docker-compose.yml ├── html │ ├── index.php │ └── phpinfo.php └── www └── index.html
Eintrag: version: "3.8" services: web: image: php:8.2-apache container_name: apache-only ports: - "8081:80" volumes: - /srv/docker/apache_php/html:/var/www/html restart: unless-stopped	pi@10.2.205.200:22 - Bitvise xterm - pi@rpi-lu... GNU nano 8.4 docker-compose.yml # docker-compose.yml version: "3.8" services: web: image: php:8.2-apache container_name: apache-only ports: - "8081:80" volumes: - /srv/docker/apache_php/html:/var/www/html restart: unless-stopped
lokales Verzeichnis gemountet	/srv/docker/apache_php/html:/var/www/html
php-Info Seite erstellt	cat > /srv/docker/apache_php/html/phpinfo.php <<'PHP' <?php phpinfo();
Falls Rückmeldung „Permission denied“ kommt → Benutzer pi Schreibrechte geben	sudo chown -R pi:pi /srv/docker/apache_php/html



9. Docker Compose Projekt

Erklärung	Beispiel
Damit WordPress den MySQL-Container erreicht, definieren wir ein externes Netzwerk, das beide Compose-Files nutzen	docker network create wpnet
zeigt sämtliche docker Netzwerke	docker network ls `pi@rpi-lugstein4:~ $ docker network ls` `NETWORK ID NAME DRIVER SCOPE` `530d9d78aafc bridge bridge local` `47e84ca08847 host host local` `a9541b3350ee mysql_default bridge local` `23e8397b121f none null local` `0e8e384c15e7 wpnet bridge local`
„stack“ liegt unter /srv/docker/wordpress-stack	
Eintrag version: "3.8" services: wordpress: image: wordpress:latest container_name: wordpress restart: unless-stopped ports: - "8080:80" # Host-Port 8080 -> Container-Port 80 environment: WORDPRESS_DB_HOST: db # Name des MySQL-Services (Netzwerk!) WORDPRESS_DB_USER: wordpress WORDPRESS_DB_PASSWORD: wordpress WORDPRESS_DB_NAME: wordpress volumes: - wordpress_data:/var/www/html # Persistente WP-Daten depends_on: - db db: image: mysql:8.0 container_name: wordpress_db restart: unless-stopped environment: MYSQL_ROOT_PASSWORD: rootpassword MYSQL_DATABASE: wordpress MYSQL_USER: wordpress MYSQL_PASSWORD: wordpress command: '--default-authentication-plugin=mysql_native_password' volumes: - db_data:/var/lib/mysql # Persistente Datenbank volumes: wordpress_data: db_data:	pi@10.2.205.200:22 - Bitvise xterm - pi@rpi-lugstein4: /srv/docker/wordpress-stack GNU nano 8.4 docker-compose.yml * `version: "3.8"` `services:` `wordpress:` `image: wordpress:latest` `container_name: wordpress` `restart: unless-stopped` `ports:` `- "8080:80" # Host-Port 8080 -> Container-Port 80` `environment:` `WORDPRESS_DB_HOST: db # Name des MySQL-Services (Netzwerk!)` `WORDPRESS_DB_USER: wordpress` `WORDPRESS_DB_PASSWORD: wordpress` `WORDPRESS_DB_NAME: wordpress` `volumes:` `- wordpress_data:/var/www/html # Persistente WP-Daten (Uploads, Themes, PLugins)` `depends_on:` `- db` `db:` `image: mysql:8.0` `container_name: wordpress_db` `restart: unless-stopped` `environment:` `MYSQL_ROOT_PASSWORD: rootpassword` `MYSQL_DATABASE: wordpress` `MYSQL_USER: wordpress` `MYSQL_PASSWORD: wordpress` `command: '--default-authentication-plugin=mysql_native_password'` `volumes:` `- db_data:/var/lib/mysql # Persistente Datenbank` `volumes:` `wordpress_data:` `db_data:` Aufruf **http://10.2.205.200:8080** 

10. Docker-Dokumentation

Erklärung	Beispiel
sollte alle container starten → Zur Sicherheit in jeden Ordner wechseln und mit docker compose up -d einzeln starten	docker start $(docker ps -a -q)
docker ps	

```
pi@rpi-lugstein4:/srv/docker $ docker ps
CONTAINER ID   IMAGE                                COMMAND
0ba66655af4c   portainer/portainer-ce:latest        "/portainer"
3751a22e7c90   php:8.2-apache                       "docker-php-entrypoi…"
a734b493c2b9   wordpress:latest                     "docker-entrypoint.s…"
b8214c3c611f   mysql:8.0                            "docker-entrypoint.s…"
5ba923d43f8b   mariadb:10.11                        "docker-entrypoint.s…"
```

```
CREATED              STATUS              PORTS                                                                NAMES
23 seconds ago       Up 22 seconds       8000/tcp, 9000/tcp, 0.0.0.0:9443->9443/tcp, [::]:9443->9443/tcp      portainer
About a minute ago   Up About a minute   0.0.0.0:8081->80/tcp, [::]:8081->80/tcp                              apache-only
26 minutes ago       Up 26 minutes       0.0.0.0:8080->80/tcp, [::]:8080->80/tcp                              wordpress
26 minutes ago       Up 26 minutes       3306/tcp, 33060/tcp                                                  wordpress_db
6 days ago           Up 5 days           3306/tcp                                                             mysql
```

docker images → **zeigt alle Images**

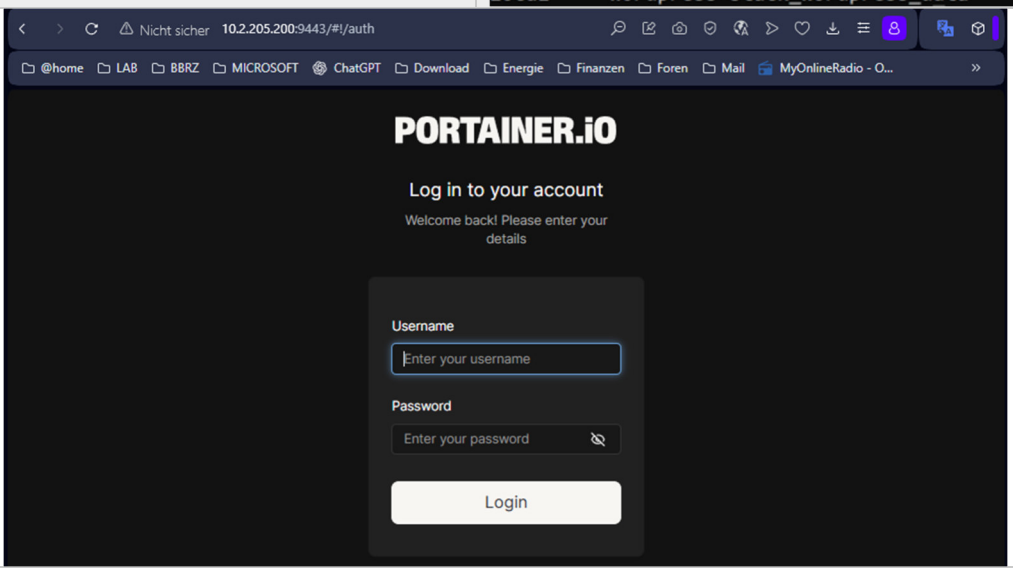
```
IMAGE                          ID             DISK USAGE   CONTENT SIZE   EXTRA
apache_php-web:latest          00e446d97792       714MB          169MB
hello-world:latest             56433a6be3fd      16.9kB         4.62kB    U
mariadb:10.11                  dbe56e20372f       433MB         99.8MB    U
mysql:8.0                      f37951fc3753      1.06GB          230MB    U
nginx:latest                   1beed3ca46ac       244MB         58.3MB    U
php:8.2-apache                 d2ba3ab5dcb2       714MB          168MB    U
portainer/portainer-ce:latest  d38a6876b61d       238MB         57.4MB    U
wordpress:latest               c9f42b6d7789      1.05GB          254MB    U
```

docker network ls → **zeigt Netzwerke**

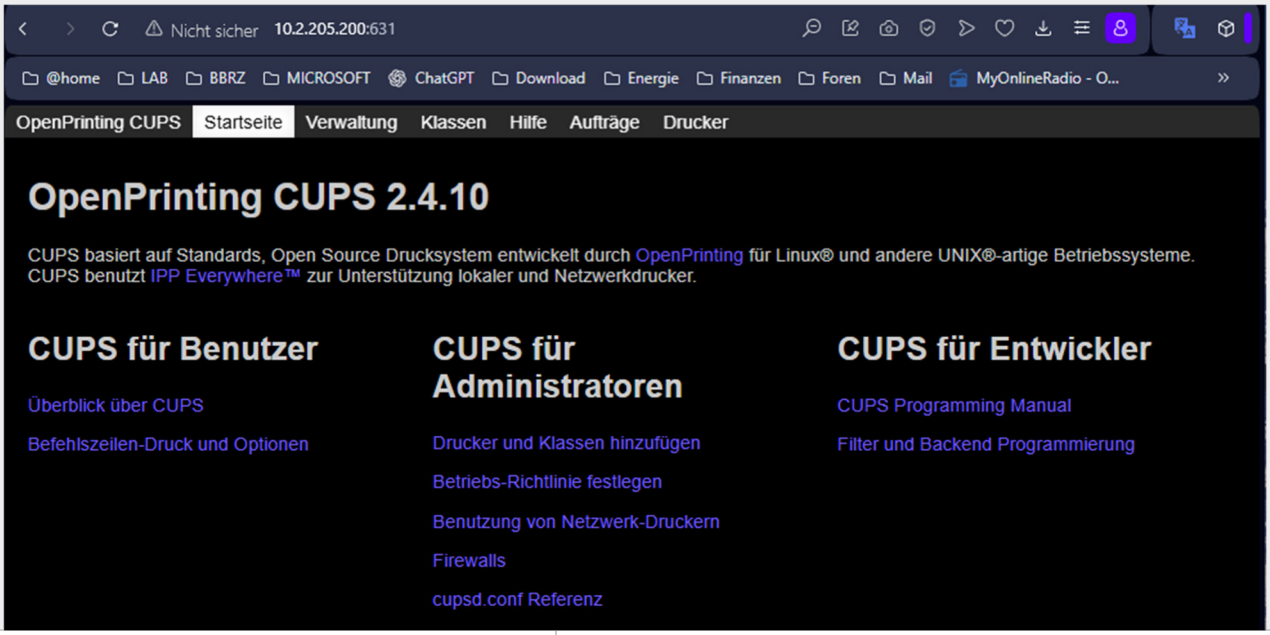
```
pi@rpi-lugstein4:/srv/docker/nginx $ docker network ls
NETWORK ID     NAME                      DRIVER    SCOPE
eb7a6726b583   apache_php_default        bridge    local
530d9d78aafc   bridge                    bridge    local
47e84ca08847   host                      host      local
a9541b3350ee   mysql_default             bridge    local
24a3931cc39c   nginx_default             bridge    local
23e8397b121f   none                      null      local
80a7a9f67358   portainer_default         bridge    local
ea15ed2a69d6   wordpress-stack_default   bridge    local
0e8e384c15e7   wpnet                     bridge    local
```

docker volume ls → **zeigt alle Volumes**

```
pi@rpi-lugstein4:/srv/docker/nginx $ docker volume ls
DRIVER    VOLUME NAME
local     wordpress-stack_db_data
local     wordpress-stack_wordpress_data
```

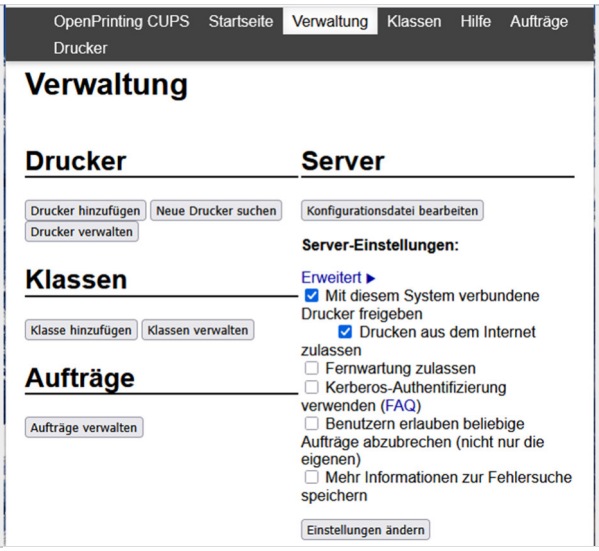


11. Drucker (CUPS)

System aktualisieren	sudo apt update
CUPS installieren	sudo apt install cups -y
Meinen Benutzer der Druckergruppe hinzufügen	sudo usermod -aG lpadmin $USER
CUPS-Dienst starten + aktivieren **➔ Automatischer Start beim booten**	sudo systemctl enable cups sudo systemctl start cups
status von CUPS abfragen	systemctl status cups
CUPS Webinterface aktivieren	sudo cupsctl WebInterface=yes
Zugriff von WIN11 auf GUI nicht möglich!	
Lauscht CUPS überhaupt auf Port 631?	sudo ss -tulpn | grep 631
Netzwerkzugriff von außen erlauben	sudo cupsctl --remote-any
Dann neu starten	sudo systemctl restart cups
Zugriff: http://10.2.205.200:631	
	
Drucker hinzufügen (simulieren eines PDF-Printers)	
	sudo apt update
	sudo apt install cups-pdf -y
Installation per GUI	

Admin Anmeldung
pi / Aa_123456

→ Verwaltung → **Drucker hinzufügen**

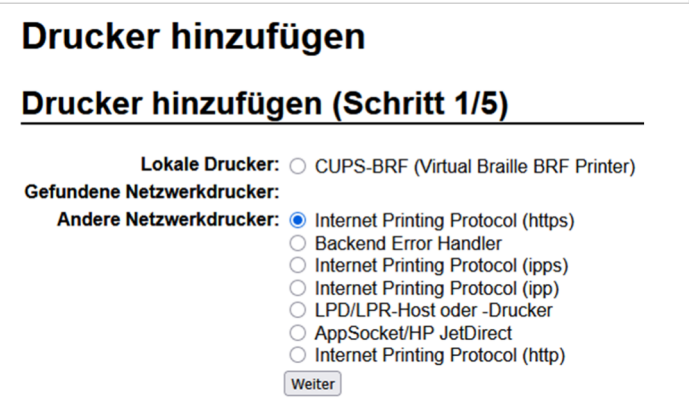


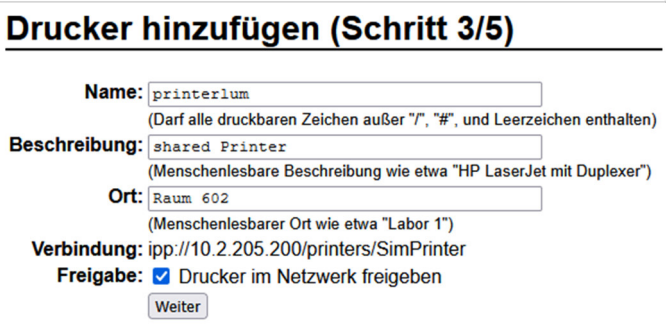
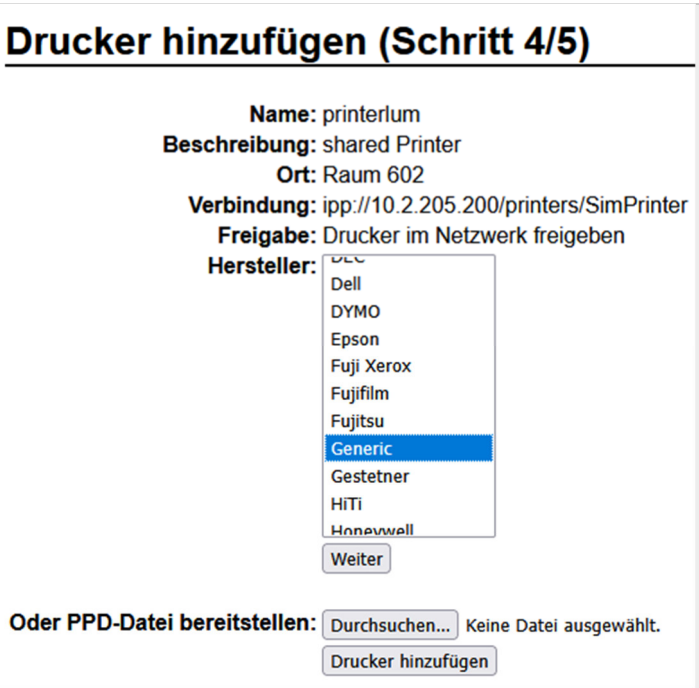
➔ **CUPS_BRF….**

hier sollte eigentlich ein virtueller .pdf Printer erscheinen → workaround

Internet Printing Protocol (https) auswählen.

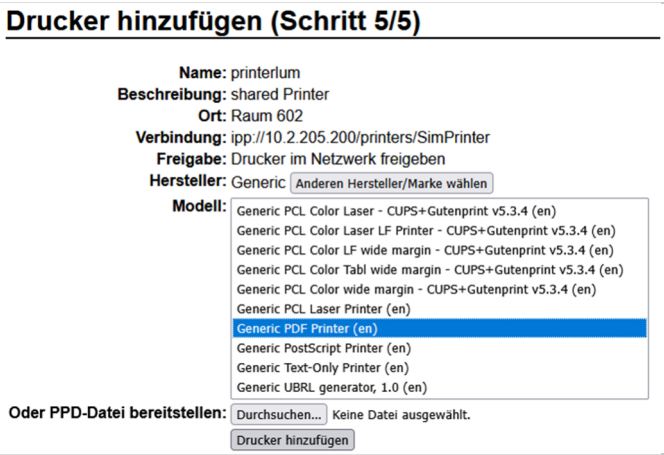
➔ **Weiter**



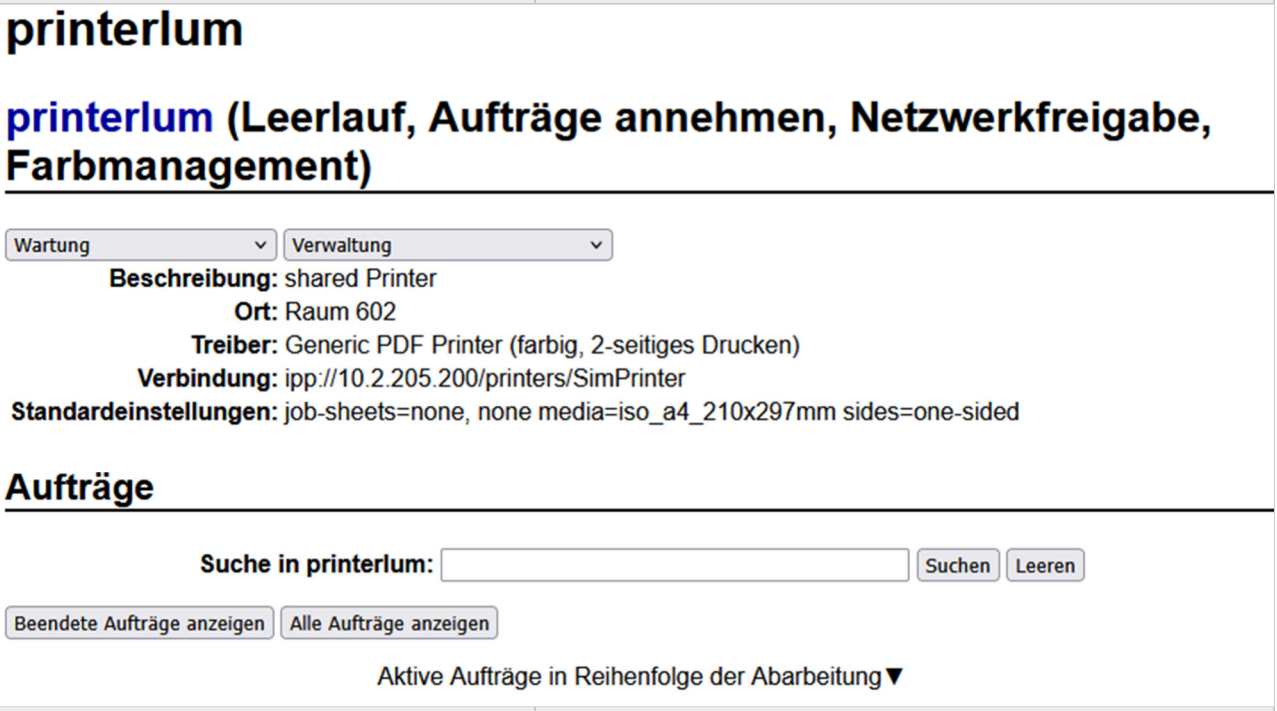
ipp://10.2.205.200/printers/SimPrinter	
	
Auswahl eines „Generic“ Druckers. ➔ **Drucker hinzufügen**	

ich entscheide mich für einen Generic PDF Printer

➔ **Drucker hinzufügen**

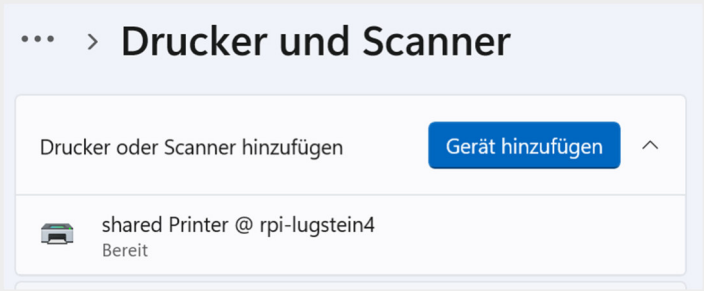


Unter Verwaltung → Drucker verwalten erscheint der Drucker



Installation des Druckers auf einem WIN11-Client

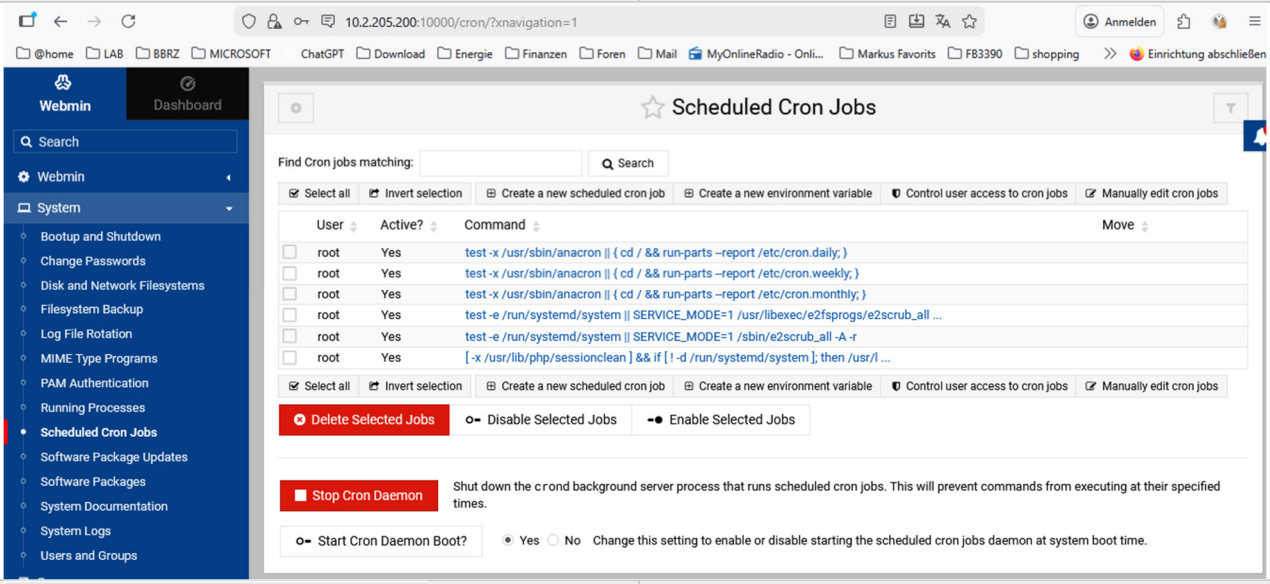
-Windows-Taste
-Einstellungen öffnen
-Bluetooth & Geräte
-Drucker & Scanner
-Drucker oder Scanner hinzufügen
-Warten bis Windows fertig gesucht hat
-Ganz unten auf → Gerät hinzufügen
Treiber wird installiert, und nach kurzer Zeit wird der Drucker als „Bereit“ angezeigt.



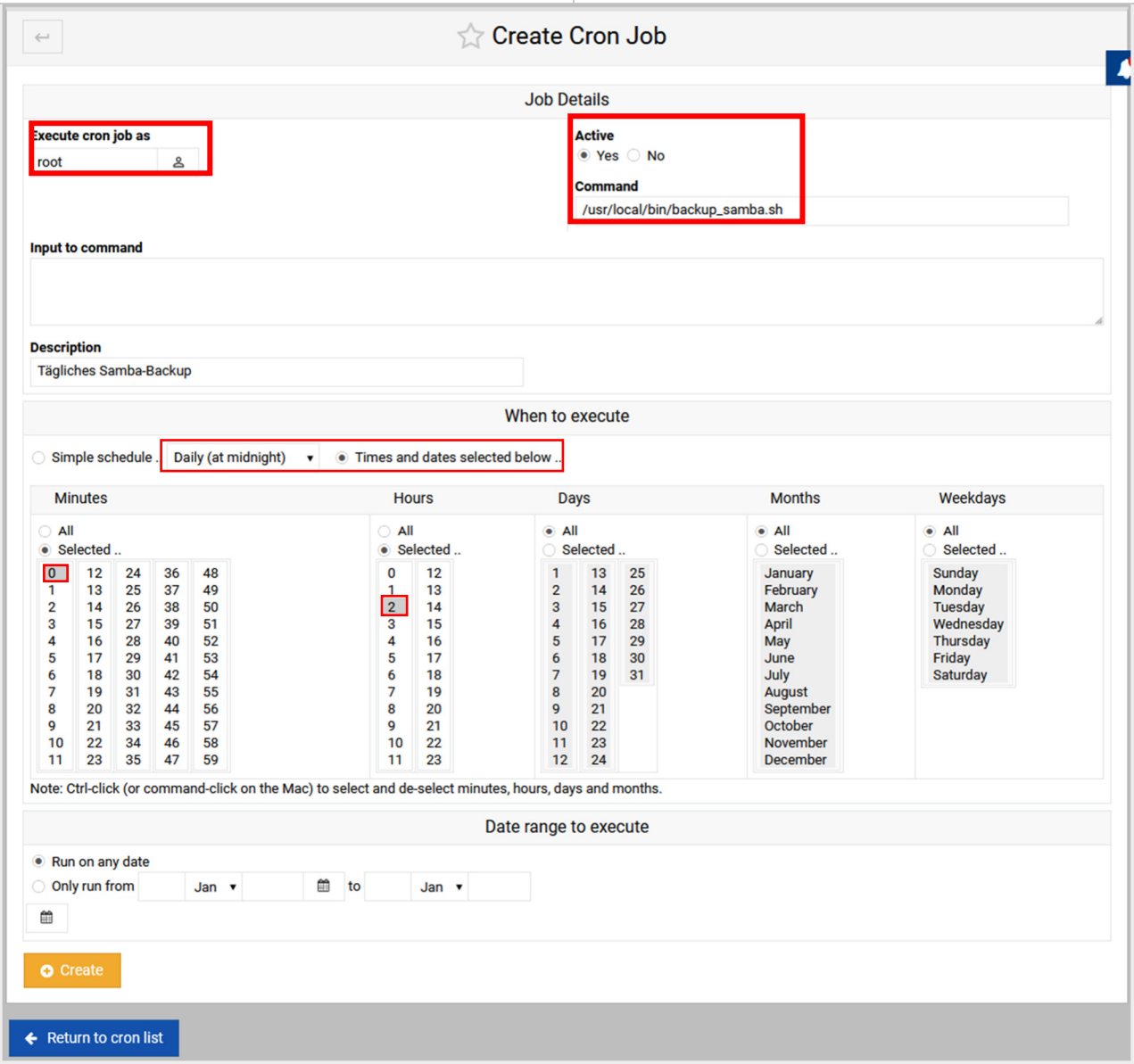
12. Cronjobs

Dafür nutze ich Webmin

12.1 Tägliches Backup der Samba-Freigaben

Erklärung	Beispiel
Unter System → Scheduled Cron Jobs	
	
Create a new scheduled cron job	
die Samba Freigaben liegen unter: /home/	
Neues Backup-Skript für deine Struktur	sudo nano /usr/local/bin/backup_samba.sh
Inhalt des skripts: **Wir sichern explizit die Home-Verzeichnisse der Nutzer, NICHT das ganze System.**	#!/bin/bash # Zielordner für Backups TARGET="/backup/samba" mkdir -p "$TARGET" # Datum für den Dateinamen DATE=$(date +%F) ARCHIVE="$TARGET/samba-backup-$DATE.tar.gz" # Zu sichernde Verzeichnisse (deine Samba-Freigaben): SOURCES=" /home/admin /home/lehrer1 /home/lehrer2 /home/schueler1 /home/schueler2 /home/schueler3 " # 1) Backup erstellen tar -czf "$ARCHIVE" $SOURCES # 2) Backups löschen, die älter als 7 Tage sind find "$TARGET" -type f -name "samba-backup-*.tar.gz" -mtime +7 -delete

ausführbar machen	sudo chmod +x /usr/local/bin/backup_samba.sh
skript ausführen	sudo /usr/local/bin/backup_samba.sh ls -l /backup/samba



System → Scheduled Cron Jobs Execute cron job as: root Active: Yes Command: /usr/local/bin/backup_samba.sh Beschreibung: z.B. Tägliches Samba-Backup (7 Tage Rotation)	When to execute Simple schedule: Daily Minuten: „Selected..“ → 0 Stunden: „Selected..“ → 2 Tage, Monate, Wochentage: jeweils „All“ Speichern.

12.2 Wöchentliche Bereinigung von Temp-Dateien

Erklärung	Beispiel
Shell-Skript erstellen	sudo nano /usr/local/bin/cleanup_tmp.sh
	#!/bin/bash # Verzeichnis, das bereinigt werden soll TARGET="/tmp" # Logs optional (kannst du auch weglassen) LOGFILE="/var/log/cleanup_tmp.log" # Datum für Log DATE="$(date '+%Y-%m-%d %H:%M:%S')" # 1) Dateien älter als 30 Tage löschen find "$TARGET" -type f -mtime +30 -print -delete >> "$LOGFILE" 2>&1 # 2) Leere Verzeichnisse älter als 30 Tage löschen (optional) find "$TARGET" -type d -empty -mtime +30 -print -delete >> "$LOGFILE" 2>&1 echo "[$DATE] Cleanup in $TARGET abgeschlossen" >> "$LOGFILE"
ausführbar machen	sudo chmod +x /usr/local/bin/cleanup_tmp.sh
Root-Crontab öffnen	sudo crontab -e
Cronjob – jeden Sonntag um 03:00 Uhr	0 3 * * 0 /usr/local/bin/cleanup_tmp.sh Bedeutung: • Minute: 0 • Stunde: 3 → 03:00 Uhr • Tag im Monat: * (egal) • Monat: * (egal) • Wochentag: 0 → Sonntag
testweise das skript starten	sudo /usr/local/bin/cleanup_tmp.sh

12.3 System-Update Benachrichtigung (Log-Datei)

Erklärung	Beispiel
skrip erstellen	sudo nano /usr/local/bin/check_updates.sh
Eintrag	#!/bin/bash LOGFILE="/var/log/system-updates.log" DATE="$(date '+%Y-%m-%d %H:%M:%S')" HOSTNAME="$(hostname)" # Paketinformationen aktualisieren

	apt update -qq # Liste der verfügbaren Updates holen (alles ab Zeile 2) UPDATES=$(apt list --upgradable 2>/dev/null \| tail -n +2) if [-z "$UPDATES"]; then MSG="$DATE [$HOSTNAME] - Keine Updates verfügbar." else MSG="$DATE [$HOSTNAME] - Updates verfügbar:\n$UPDATES" fi # In Log-Datei schreiben echo -e "$MSG\n" >> "$LOGFILE"
ausführbar machen	sudo chmod +x /usr/local/bin/check_updates.sh
einmal manuell ausführen	sudo /usr/local/bin/check_updates.sh
prüfen	sudo tail -n 30 /var/log/system-updates.log
	pi@rpi-lugstein4:/home $ sudo tail -n 30 /var/log/system-updates.log libsystemd0/stable 257.9-1~deb13u1 arm64 [upgradable from: 257.8-1~deb13u2] libtalloc2/stable 2:2.4.3+samba4.22.6+dfsg-0+deb13u1 arm64 [upgradable from: 2:2.4.3+samba4.22.4+dfsg-1~deb13u1] libtdb1/stable 2:1.4.13+samba4.22.6+dfsg-0+deb13u1 arm64 [upgradable from: 2:1.4.13+samba4.22.4+dfsg-1~deb13u1] libtevent0t64/stable 2:0.16.2+samba4.22.6+dfsg-0+deb13u1 arm64 [upgradable from: 2:0.16.2+samba4.22.4+dfsg-1~deb13u1]
Root-Crontab öffnen	sudo crontab -e
Cronjob – jeden Montag um 08:00 Uhr	0 8 * * 1 /usr/local/bin/check_updates.sh Bedeutung: • 0 Minute • 8 Uhr • * * jeder Tag / jeder Monat • 1 = Montag

12.4 Docker Container Gesundheitscheck

	sudo nano /usr/local/bin/docker_healthcheck.sh
	#!/bin/bash LOGFILE="/var/log/docker-health.log" DATE="$(date '+%Y-%m-%d %H:%M:%S')" # Alle Container (auch gestoppte) ALL_CONTAINERS=$(docker ps -a --format '{{.Names}} {{.Status}}') # Nur laufende Container RUNNING_CONTAINERS=$(docker ps --format '{{.Names}}') # Standardmeldung echo "[$DATE] Docker Healthcheck gestartet" >> "$LOGFILE" if [-z "$ALL_CONTAINERS"]; then

	echo "[$DATE] Keine Container vorhanden." >> "$LOGFILE" exit 0 fi # Status aller Container loggen echo "[$DATE] Status aller Container:" >> "$LOGFILE" echo "$ALL_CONTAINERS" >> "$LOGFILE" # Prüfen, ob es Container gibt, die NICHT laufen # (Status beginnt dann nicht mit 'Up') NOT_RUNNING=$(docker ps -a --format '{{.Names}} {{.Status}}' \| grep -v '^.* Up ') if [-z "$NOT_RUNNING"]; then echo "[$DATE] ALLE Container laufen." >> "$LOGFILE" else echo "[$DATE] NICHT laufende Container gefunden:" >> "$LOGFILE" echo "$NOT_RUNNING" >> "$LOGFILE" fi echo "" >> "$LOGFILE"
	sudo chmod +x /usr/local/bin/docker_healthcheck.sh
manuell ausführen	sudo /usr/local/bin/docker_healthcheck.sh
log prüfen	sudo tail -n 50 /var/log/docker-health.log
	pi@rpi-lugstein4:/home $ sudo tail -n 50 /var/log/docker-health.log [2025-11-17 12:27:08] Docker Healthcheck gestartet [2025-11-17 12:27:08] Status aller Container: nginx-web-1 Up 4 hours portainer Up 4 hours apache-only Up 4 hours wordpress Up 4 hours wordpress_db Up 4 hours mysql Up 5 days romantic_joliot Exited (0) 4 hours ago [2025-11-17 12:27:08] NICHT laufende Container gefunden: romantic_joliot Exited (0) 4 hours ago
	sudo crontab -e
	0 */6 * * * /usr/local/bin/docker_healthcheck.sh Bedeutung: • Minute `0` • jede 6. Stunde (`*/6` → 0, 6, 12, 18 Uhr) • jeden Tag, jeden Monat, jeder Wochentag

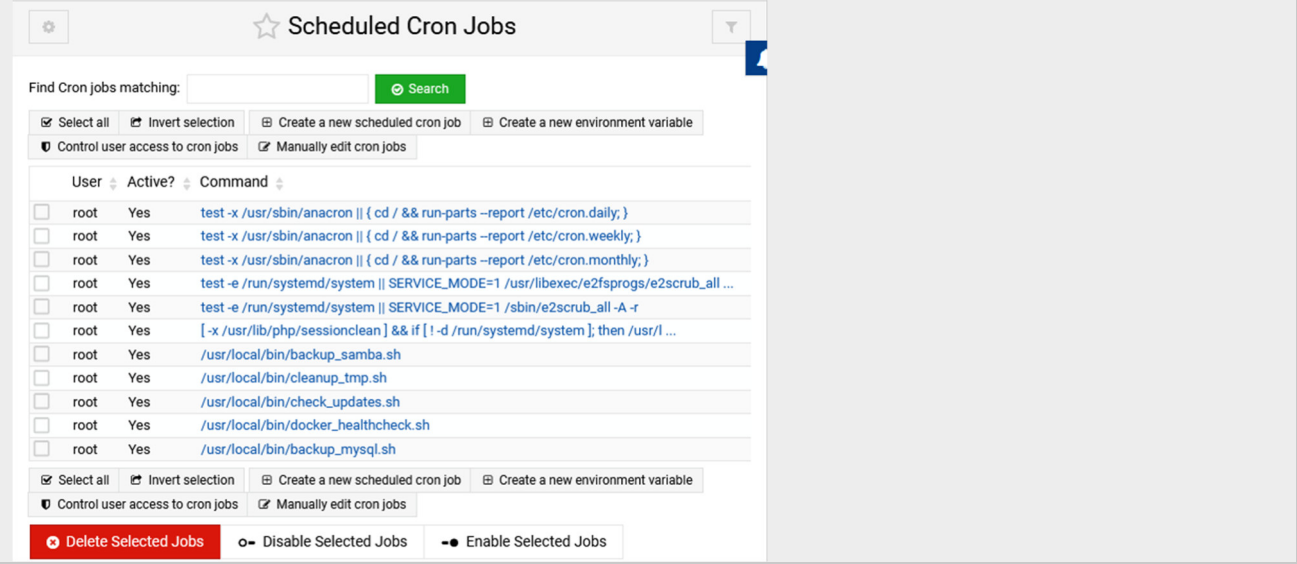
12.5 Datenbank-Backup

Erklärung	Beispiel
Damit mysqldump ohne Passwort-Abfrage läuft, legst du für root eine Config an.	sudo nano /root/.my.cnf
Eintrag in die .my.cnf	[client] user=root

	password=Aa_123456
Rechte einschränken	sudo chmod 600 /root/.my.cnf
	sudo mysqldump --all-databases > /tmp/test.sql
Backup-Skript erstellen	sudo mkdir -p /backup/mysql
Skript anlegen	sudo nano /usr/local/bin/backup_mysql.sh
	#!/bin/bash # Zielverzeichnis für Backups TARGET="/backup/mysql" mkdir -p "$TARGET" # Zeitstempel DATE="$(date +%F_%H-%M)" # Dateiname ARCHIVE="$TARGET/mysql-backup-$DATE.sql.gz" # Alle Datenbanken sichern und komprimieren mysqldump --all-databases --single-transaction --quick --lock-tables=false | gzip > "$ARCHIVE"
ausführbar machen	sudo chmod +x /usr/local/bin/backup_mysql.sh
Test: Backup manuell ausführen	sudo /usr/local/bin/backup_mysql.sh
	ls -l /backup/mysql

```
pi@rpi-lugstein4:/home $ ls -l /backup/mysql
total 1040
-rw-r--r-- 1 root root 530985 Nov 17 12:47 mysql-backup-2025-11-17_12-47.sql.gz
-rw-r--r-- 1 root root 530985 Nov 17 12:48 mysql-backup-2025-11-17_12-48.sql.gz
```

Testlauf	sudo /usr/local/bin/backup_mysql.sh ls -l /backup/mysql
	sudo crontab -e
täglich um 1:00 Uhr	0 1 * * * /usr/local/bin/backup_mysql.sh Bedeutung: • Minute: 0 • Stunde: 1 → 01:00 Uhr • Tag: * (jeden Tag) • Monat: * • Wochentag: *



12.6 UFW-Firewall

UFW installieren & aktivieren	sudo apt update sudo apt install ufw -y
Standardeinstellungen setzen	sudo ufw default deny incoming sudo ufw default allow outgoing
SSH erlauben → sonst Sperre!	sudo ufw allow ssh
Cockpit erlauben (TCP-Port 9090)	sudo ufw allow 9090/tcp
Webmin erlauben (TCP-Port 10000)	sudo ufw allow 10000/tcp
HTTP + HTTPS erlauben (Webserver + WordPress)	sudo ufw allow http sudo ufw allow https
Samba erlauben **verwendet mehrere Ports** **oder** sudo ufw allow samba	sudo ufw allow 137/udp sudo ufw allow 138/udp sudo ufw allow 139/tcp sudo ufw allow 445/tcp
FTP erlauben	sudo ufw allow 21/tcp
MySQL nur lokal erlauben (default)	sudo ufw allow from 127.0.0.1 to any port 3306 sudo ufw allow from ::1 to any port 3306
Docker-Ports erlauben	
WordPress (8080)	sudo ufw allow 8080/tcp
Apache-PHP (8081)	sudo ufw allow 8081/tcp
Nginx Test (8082)	sudo ufw allow 8082/tcp
Portainer UI (9443)	sudo ufw allow 9443/tcp
Optional: Portainer Agent Ports	sudo ufw allow 8000/tcp sudo ufw allow 9000/tcp
BEIDE MySQL-Container sollen NICHT aus dem Netzwerk erreichbar sein.	sudo ufw deny 3306
Falls lokal (127.0.0.1) erlaubt sein soll:	sudo ufw allow from 127.0.0.1 to any port 3306 sudo ufw allow from ::1 to any port 3306
Firewall aktivieren	sudo ufw enable
Status prüfen	sudo ufw status verbose

```
pi@rpi-lugstein4:/home $ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), deny (outgoing), deny (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
9090/tcp                   ALLOW IN    Anywhere
10000/tcp                  ALLOW IN    Anywhere
80/tcp                     ALLOW IN    Anywhere
443                        ALLOW IN    Anywhere
137/udp                    ALLOW IN    Anywhere
138/udp                    ALLOW IN    Anywhere
139/udp                    ALLOW IN    Anywhere
445/udp                    ALLOW IN    Anywhere
21/tcp                     ALLOW IN    Anywhere
3306                       ALLOW IN    127.0.0.1
8080/tcp                   ALLOW IN    Anywhere
8081/tcp                   ALLOW IN    Anywhere
8082/tcp                   ALLOW IN    Anywhere
9443/tcp                   ALLOW IN    Anywhere
8000/tcp                   ALLOW IN    Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)
9090/tcp (v6)              ALLOW IN    Anywhere (v6)
10000/tcp (v6)             ALLOW IN    Anywhere (v6)
80/tcp (v6)                ALLOW IN    Anywhere (v6)
443 (v6)                   ALLOW IN    Anywhere (v6)
137/udp (v6)               ALLOW IN    Anywhere (v6)
138/udp (v6)               ALLOW IN    Anywhere (v6)
139/udp (v6)               ALLOW IN    Anywhere (v6)
445/udp (v6)               ALLOW IN    Anywhere (v6)
21/tcp (v6)                ALLOW IN    Anywhere (v6)
3306                       ALLOW IN    ::1
8080/tcp (v6)              ALLOW IN    Anywhere (v6)
8081/tcp (v6)              ALLOW IN    Anywhere (v6)
8082/tcp (v6)              ALLOW IN    Anywhere (v6)
9443/tcp (v6)              ALLOW IN    Anywhere (v6)
8000/tcp (v6)              ALLOW IN    Anywhere (v6)
```