

35_Docker Compose mit Ubuntu Server

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

Inhalt

1. Zielsetzung	2
2. Geräteübersicht	2
2.1 Netzwerkeinstellungen (Hardware)	2
2.2 User & Zugangsdaten (Software)	2
3. Installation einer VM (Hyper-V)	3
4. Installation Ubuntu-Server	5
5. docker & Tools	7
5.1 Installation von Tools	7
5.2 Installation cockpit	7
5.3 Prüfung der docker-Versionen	8
5.4 Wichtige Info zu docker!!	8
5.5 Docker Kommandos/Befehle	9
6. Beispiele mit „docker“	9
6.1 Einfacher Webserver mit Nginx	9
6.2 Einfacher Webserver mit „Apache“ und Startseite	10
6.3 Apache, mySQL, phpMyAdmin und php-Datei	11
6.3.1 Schnellfix: PHP anpassen und Tabelle bei Bedarf anlegen.....	12
6.4 Mit Datenbank (MySQL & Adminer)	14
6.5 Wordpress & MySQL	17
6.6 Debian & Cowsay	18

1. Zielsetzung

Ziel ist die Einrichtung und Vernetzung virtueller Maschinen mit gemeinsamen Freigaben, Fernzugriff und einem automatisierten Cron-Job zur Systemüberwachung.

2. Geräteübersicht

	Device (# & Name)	Rolle	Hypervisor (VM)	Betriebssystem (OS)	IP-Adresse (LAN)
1	LENOVO E16	Host-PC	physikalisch	WIN11 Pro	
2					
3					

2.1 Netzwerkeinstellungen (Hardware)

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

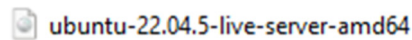
2.2 User & Zugangsdaten (Software)

Client	Geräte #	Anwendung	Benutzername	Passwort	Gerätename	domain (AD)
Server	Geräte #	Anwendung	Benutzername	Passwort	Hostname	

3. Installation einer VM (Hyper-V)

Installation eines UBUNTU-SERVERS von <https://releases.ubuntu.com/jammy>

Der download liegt als ISO-Datei vor.



Erklärung	Beispiel
Öffnen der Virtualisierungs-Software „Hyper-V-Manager“	
Netzwerkkarte „extern“ erstellen und (bridge) der VM zuweisen. → LUM_LAP → Anwenden & OK	
Auf der rechten Seite im oberen Bereich erstellen wir durch einen Klick auf „NEU“ einen neuen virtuellen Computer mit einem WIN11-Image. Name: VM_UBUNTU_LAP	
	Es öffnet sich der Assistent in den wir jetzt Schritt für Schritt unsere Daten für die VM eintragen.

Vorbemerkungen	Sie können das Betriebssystem jetzt installieren, sofern Ihnen die erforderlichen Setupmedien zur Verfügung stehen, oder diesen Vorgang zu einem späteren Zeitpunkt ausführen.
Name und Pfad angeben	☐ Betriebssystem zu einem späteren Zeitpunkt installieren
Generation angeben	☒ Betriebssystem von einer startbaren CD/DVD-ROM installieren
Speicher zuweisen	Medien
Netzwerk konfigurieren	☐ Physisches CD/DVD-Laufwerk:
Virtuelle Festplatte verbinden	☒ Abbilddatei (ISO): ubuntu-22.04.5-live-server-amd64.iso Durchsuchen...
Installationsoptionen	
Zusammenfassung	

Name und Pfad angeben: **ubuntu-22.04.5_markus_Aa_123456**

Generation angeben: **2.Generation**

Speicher zuweisen: 4096 MB

Netzwerk konfigurieren: NIC_LAP

Virtuelle Festplatte verbinden: virtuelle Festplatte erstellen

Größe: 60GB

Installationsoptionen: Betriebssystem von einem startbaren Medium installieren → Abbilddatei (ISO) → Speicherort der ISO-Datei!!!

Zusammenfassung: → Fertigstellen

Vorbemerkungen

Name und Pfad angeben

Generation angeben

Speicher zuweisen

Netzwerk konfigurieren

Virtuelle Festplatte verbinden

Installationsoptionen

Zusammenfassung

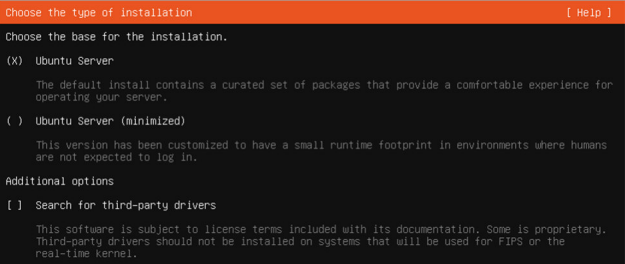
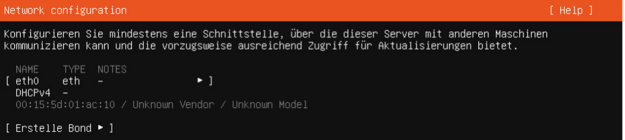
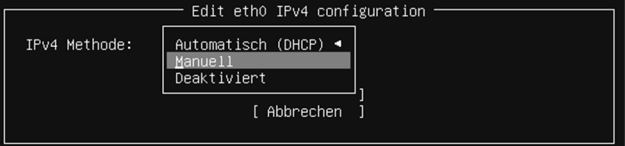
Name:	ubuntu-22.04.5_markus_Aa_123456
Generation:	Generation 2
Arbeitsspeicher:	4096 MB
Netzwerk:	NIC_LAP
Festplatte:	C:\ProgramData\Microsoft\Windows\Virtual Hard Disks\ubuntu-22.04.5_markus_Aa_
Betriebssystem:	Installation von "C:\Images\Linux\Ubuntu\ubuntu-22.04.5-live-server-amd64.iso"

Die VM wird erstellt und erscheint in der Liste der VM's!

Wichtig:

ubuntu-22.04.5_markus_Aa_123456	◀ ▶ ↺
Hardware	Sicherheit
Hardware hinzufügen	Sicherer Start
Firmware	Verwenden Sie "Sicherer Start", um zu verhindern, dass nicht autorisierter Code zur Startzeit ausgeführt wird (empfohlen).
Von "DVD-Laufwerk" starten	☐ Sicheren Start aktivieren
Sicherheit	
"Sicherer Start" ist deaktiviert	

4. Installation Ubuntu-Server

Erklärung	Beispiel
Ubuntu Server installieren	
Manuelle (statische) IP direkt bei der Installation vergeben!	
IPv4 Methode ändern	
→ umstellen auf manuell	
Subnetz: 192.168.0.0/24 Adresse: 192.168.0.125 Gateway: 192.168.0.254 Nameserver: 8.8.8.8	
statische IP ist gesetzt!	
Da der Server ohne GUI ist, bevorzuge ich die Installation von SSH sicherheitshalber	

Wähle nur **docker** aus und drücke Enter
Damit bekommst du das offizielle Docker
Snap-Paket, das sofort lauffähig ist,
und du kannst danach direkt mit **docker**
compose oder **docker-compose**
weitermachen!

Featured server snaps			[Help]
Dies sind beliebte Snaps in Serverumgebungen. Aktivieren oder deaktivieren Sie die Auswahl mit SPACE, drücken Sie ENTER, um weitere Details zu Paket, Herausgeber und verfügbaren Versionen anzuzeigen.			
[]	microk8s	canonical✓	Kubernetes for workstations and appliances
[]	nextcloud	nextcloud✓	Nextcloud Server - A safe home for all your data
[]	wekan	xet7	Open-Source kanban
[]	kata-containers	katacontainers✓	Build lightweight VMs that seamlessly plug into the co
[]	docker	canonical✓	Docker container runtime
[]	canonical-livepatch	canonical✓	Canonical Livepatch Client
[]	rocketchat-server	rocketchat✓	Rocket.Chat server
[]	mosquitto	mosquitto✓	Eclipse Mosquitto MQTT broker
[]	etcd	canonical✓	Resilient key-value store by CoreOS
[]	powershell	canonical✓	Powershell for every system!
[]	sabnzbd	safihre	SABnzbd
[]	wormhole	snappcrafters✓	get things from one computer to another, safely
[]	aws-cli	aws	Universal Command Line Interface for Amazon Web Servic
[]	google-cloud-sdk	google-cloud-sdk✓	Google Cloud SDK
[]	slcli	softlayer	Python based SoftLayer API Tool.
[]	doctl	digitalocean✓	The official DigitalOcean command line interface
[]	postgres10	cmd✓	PostgreSQL is a powerful, open source object-relations
[]	keepalived	keepalived-project✓	High availability VRRP/BFD and load-balancing for Linu
[]	prometheus	canonical✓	The Prometheus monitoring system and time series datab

```
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-157-generic x86_64)
```

```
* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/pro
```

```
System information as of Mo 13. Okt 14:20:55 UTC 2025
```

```
System load:            0.14
Usage of /:              19.3% of 27.86GB
Memory usage:           8%
Swap usage:             0%
Processes:              162
Users logged in:        0
IPv4 address for eth0:  192.168.0.123
IPv6 address for eth0:  fd81:691c:d1d7:0:215:5dff:fe01:ac11
IPv6 address for eth0:  2a02:1748:dd4d:d990:215:5dff:fe01:ac11
```

```
Erweiterte Sicherheitswartung (ESM) für Applications ist nicht aktiviert.
```

```
60 Aktualisierungen können sofort angewendet werden.
```

```
Zum Anzeigen dieser zusätzlichen Aktualisierungen bitte »apt list --upgradable« ausführen
```

```
Aktivieren Sie ESM Apps, um zusätzliche zukünftige Sicherheitsupdates zu erhalten.
Siehe https://ubuntu.com/esm oder führen Sie aus: sudo pro status
```

```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

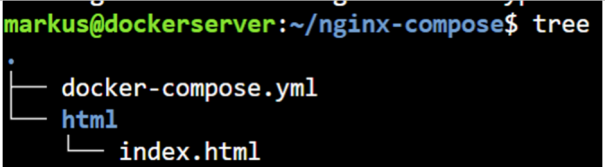
```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

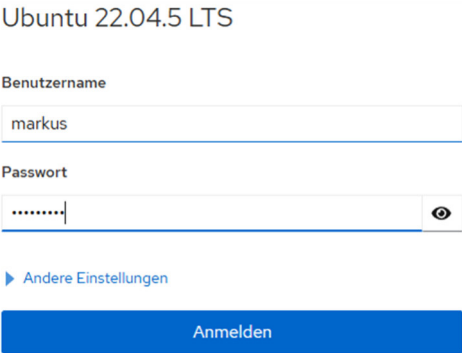
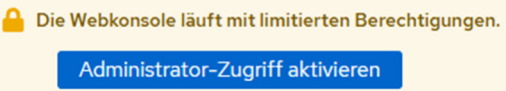
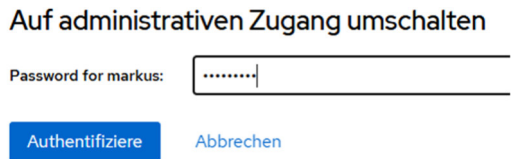
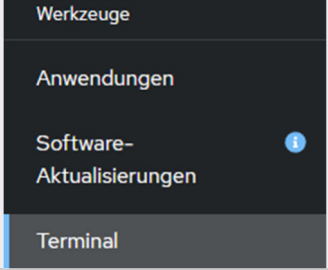
```
markus@dockerserver:~$ _
```

5. docker & Tools

5.1 Installation von Tools

Erklärung	Beispiel
Um später die Verzeichnisstruktur in Ubuntu anzuzeigen, kannst du den Befehl tree verwenden. Dazu aber muss man „tree“ erst installieren →	<pre>sudo apt update</pre> <pre>sudo apt install tree</pre>
mit dem Befehl „tree“ wird die aktuelle Ordnerstruktur angezeigt!	 <pre>markus@dockerserver:~/nginx-compose\$ tree . ├── docker-compose.yml ├── html └── index.html</pre>
später sehr hilfreich!	

5.2 Installation cockpit

Erklärung	Beispiel
Cockpit installieren	<pre>sudo apt install cockpit</pre>
Login cockpit	
Admin-Rechte aktivieren	
Passwort für administrativen Zugang	
Terminal öffnen und sich über copy, paste und Vergrößerung freuen!	

5.3 Prüfung der Docker-Versionen

Erklärung	Beispiel
docker version prüfen	<pre>sudo docker --version</pre> <pre>markus@dockerserver:~\$ sudo docker --version</pre> <pre>Docker version 28.1.1+1, build 068a01e</pre>
docker compose version prüfen	<pre>sudo docker compose --version</pre> <pre>markus@dockerserver:~\$ sudo docker compose version</pre> <pre>Docker Compose version v2.33.1</pre>
 Das ist die moderne Snap-Version von Docker (aktuell 2025er Stand). Diese Version enthält Docker Compose bereits integriert – man braucht es nicht extra zu installieren.	
Snap installiert Docker so, dass nur Root ihn direkt nutzen darf. Damit du ihn als dein Benutzer (markus) verwenden kannst:	<pre>markus@dockerserver:~\$ sudo groupadd docker</pre> <pre>sudo usermod -aG docker \$USER</pre> <pre>newgrp dockermarkus@dockerserver:~\$</pre> <pre>markus@dockerserver:~\$</pre>
Test	<pre>sudo docker ps</pre> <pre>markus@dockerserver:~\$ sudo docker ps</pre> <pre>CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES</pre> <pre>markus@dockerserver:~\$</pre>
docker & docker compose laufen!	

5.4 Wichtige Info zu docker!!

Beide Befehle dienen demselben Zweck

Mehrere Container gleichzeitig zu verwalten – also Anwendungen mit mehreren Diensten (z. B. Webserver, Datenbank, phpMyAdmin) zu starten, zu stoppen und zu konfigurieren.

Merkmal	docker-compose (v1)	docker compose (v2)
Auslieferung	Separates Python-Tool	In Docker integriert
Installation	über apt install docker-compose oder pip	automatisch mit Docker (ab v20.10)
Befehlssyntax	docker-compose up -d	docker compose up -d
Ort des Binärprogramms	/usr/local/bin/docker-compose	Bestandteil des Docker CLI (/usr/bin/docker)
Entwicklung	Eingestellt (nur Sicherheitsupdates)	Aktiv weiterentwickelt
Leistung	Etwas langsamer	Besser integriert, schneller
Kompatibilität	Alle alten Compose-Files (v1 bis v3)	Voll kompatibel mit v3.x Dateien
Verfügbarkeit	Manuell installieren	Teil des Docker Snap oder apt-Pakets

Ab Docker 20.10 ist Compose v2 (docker compose) standardmäßig enthalten.

Die ältere Variante docker-compose gilt als veraltet und sollte nicht mehr verwendet werden.

Tipp:

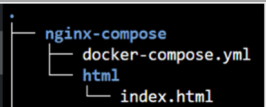
Für neue Projekte solltest du immer die moderne Syntax docker compose (ohne Bindestrich) verwenden.

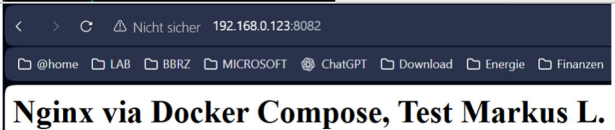
5.5 Docker Kommandos/Befehle

Befehl	Beschreibung
docker ps	Zeigt laufende Container
docker images	Listet alle lokalen Images
docker pull <image>	Lädt ein Image aus Docker Hub
docker run -d -p 8080:80 <image>	Startet einen Container im Hintergrund (z. B. Webserver)
docker stop <name>	Stoppt einen laufenden Container
docker rm <name>	Entfernt einen gestoppten Container
docker compose up -d	Startet alle Dienste aus docker-compose.yml
docker compose down	Stoppt und entfernt alle Compose-Container
docker logs -f <name>	Zeigt Live-Logs eines Containers
docker exec -it <name> bash	Öffnet eine interaktive Shell im Container

6. Beispiele mit „docker“

6.1 Einfacher Webserver mit Nginx

Erklärung	Beispiel
Ordnerstruktur erstellen	 <code>mkdir -p ~/nginx-compose/html</code>
Datei im Verzeichnis erstellen mit dem entsprechenden Eintrag	<code>echo '<h1>Nginx via Docker Compose</h1>' > ~/nginx-compose/html/index.html</code>
in das Verzeichnis wechseln	<code>cd ~/nginx-compose</code>

mit nano eine .yaml Datei erstellen und folgenden Eintrag darin machen: sudo nano docker-compose.yml <pre>version: '3.8' services: nginx: image: nginx:alpine container_name: nginx ports: - "8082:80" # Host:Container volumes: - ./html:/usr/share/nginx/html:ro restart: unless-stopped</pre>	<pre>version: '3.8' services: nginx: image: nginx:alpine container_name: nginx ports: - "8082:80" # Host:Container volumes: - ./html:/usr/share/nginx/html:ro restart: unless-stopped</pre>
container starten →	sudo docker-compose up -d <pre>[+] Running 1/1 ✓ Container nginx Created Attaching to nginx</pre>
http://192.168.0.123:8082	

6.2 Einfacher Webserver mit „Apache“ und Startseite

Erklärung	Beispiel
erstellt in der aktuellen Struktur (home) den Ordner apache-compose und den Unterordner html.	<pre>markus@dockerserver:~\$ tree ├── apache-compose │ └── html</pre> mkdir -p ~/apache-compose/html
erstellt den Eintrag „Apache läuft in Docker & Startseite von Markus“ in der Datei index.html	<pre>echo '<!doctype html><html><head><meta charset="utf-8"><title>Apache </title></head><body><h1>Apache läuft in Docker</h1><p>Startseite von Markus</p></body></html>' > ~/apache- compose/html/index.html</pre>
in den Ordner „apache-compose“ wechseln	cd ~/apache-compose
nano-Editor öffnen und folgenden Eintrag hinzufügen: <pre>version: '3.8' services: apache: image: httpd:alpine container_name: apache ports: - "8083:80" # Host:Container volumes: - ./html:/usr/local/apache2/htdocs:/ro restart: unless-stopped</pre>	<pre>GNU nano 6.2 version: '3.8' services: apache: image: httpd:alpine container_name: apache ports: - "8083:80" # Host:Container volumes: - ./html:/usr/local/apache2/htdocs:/ro restart: unless-stopped</pre>
container starten →	sudo docker-compose up -d
http://192.168.0.123:8083	

6.3 Apache, mySQL, phpMyAdmin und php-Datei

Erklärung	Beispiel
erstellt in der aktuellen Struktur (home) den Ordner lamp-compose und jeweils den Unterordner html und db	<pre>mkdir -p ~/lamp-compose/html ~/lamp-compose/db</pre> 
wechselt in den Ordner „lamp-compose/html“ eine Datei mit dem Namen „index.php“ erstellen.	<pre>cd ~/lamp-compose/html</pre> <pre>sudo touch index.php</pre>
Eintrag in der index.php Datei: <pre><?php \$dsn = "mysql:host=db;dbname=firma;charset=utf8mb4"; \$user = "app"; \$pass = "apppass"; try { \$pdo = new PDO(\$dsn, \$user, \$pass, [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]); echo "<h1>PHP + Apache + MySQL läuft! Gruß Markus L.</h1>"; echo "<p>Verbindung zur DB erfolgreich.</p>"; \$stmt = \$pdo->query("SELECT id, name, email FROM kunden ORDER BY id"); echo "<h2>Kunden</h2>"; foreach (\$stmt as \$row) { echo "#{\$row['id']} {\$row['name']} ({\$row['email']})"; } echo ""; } catch (PDOException \$e) { http_response_code(500); echo "<h1>DB-Verbindung fehlgeschlagen</h1>"; echo "<pre>". \$e->getMessage(). "</pre>"; }</pre>	<pre>sudo nano index.php</pre> 
in das Verzeichnis „lamp-compose“ wechseln und Eintrag in der docker-compose.yml:	<pre>cd lamp-compose/</pre> <pre>sudo nano docker-compose.yml</pre>

<pre> version: '3.8' services: web: image: php:8.2-apache container_name: apache_php ports: - "8080:80" volumes: - ./html:/var/www/html environment: # Für PDO MySQL (optional, nützlich für mbstring etc.) # Installierst du libs später per docker-php-ext-install, wenn gebraucht APACHE_DOCUMENT_ROOT: /var/www/html depends_on: - db restart: unless-stopped db: image: mysql:8.0 container_name: mysql_db environment: MYSQL_ROOT_PASSWORD: root MYSQL_DATABASE: firma MYSQL_USER: app MYSQL_PASSWORD: apppass volumes: - db_data:/var/lib/mysql - ./db:/docker-entrypoint-initdb.d:ro # init SQLs restart: unless-stopped phpmyadmin: image: phpmyadmin/phpmyadmin container_name: phpmyadmin ports: - "8081:80" environment: PMA_HOST: db PMA_USER: root PMA_PASSWORD: root depends_on: - db restart: unless-stopped volumes: db_data: </pre>	<pre> yaml version: '3.8' services: web: image: php:8.2-apache container_name: apache_php ports: - "8080:80" volumes: - ./html:/var/www/html environment: # Für PDO MySQL (optional, nützlich für mbstring etc.) # Installierst du libs später per docker-php-ext-install, wenn gebraucht APACHE_DOCUMENT_ROOT: /var/www/html depends_on: - db restart: unless-stopped db: image: mysql:8.0 container_name: mysql_db environment: MYSQL_ROOT_PASSWORD: root MYSQL_DATABASE: firma MYSQL_USER: app MYSQL_PASSWORD: apppass volumes: - db_data:/var/lib/mysql - ./db:/docker-entrypoint-initdb.d:ro # init SQLs restart: unless-stopped phpmyadmin: image: phpmyadmin/phpmyadmin container_name: phpmyadmin ports: - "8081:80" environment: PMA_HOST: db PMA_USER: root PMA_PASSWORD: root depends_on: - db restart: unless-stopped volumes: db_data: </pre>
start des docker-containers	sudo docker-compose up -d
Web:	http://192.168.0.123:8080 
phpMyAdmin:	http://192.168.0.123:8081 Login: root / root (oder app / apppass) DB: firma

6.3.1 Schnellfix: PHP anpassen und Tabelle bei Bedarf anlegen

Erklärung	Beispiel
Beim Aufruf der PHP-Seite erschien die Fehlermeldung: Die Verbindung zur Datenbank funktioniert, aber die Tabelle kunden existierte nicht.	
Lösung	

In der PHP-Datei (index.php) wurde ein Schnellfix eingebaut, der beim Start automatisch:

- die Tabelle kunden erstellt, falls sie fehlt,
- einen Beispiel-Datensatz („Max Mustermann“) einfügt.

```
<?php
$dns = "mysql:host=db;dbname=firma;charset=utf8mb4";

$user = "app";

$pass = "apppass";

try {

    $pdo = new PDO($dns, $user, $pass, [PDO::ATTR_ERRMODE =>
    PDO::ERRMODE_EXCEPTION]);

    $pdo->exec("

CREATE TABLE IF NOT EXISTS kunden (

    id INT AUTO_INCREMENT PRIMARY KEY,

    vorname VARCHAR(100) NOT NULL,

    nachname VARCHAR(100) NOT NULL,

    email VARCHAR(190) UNIQUE,

    erstellt_am TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);

");

    $pdo->exec("

INSERT INTO kunden (vorname, nachname, email)

SELECT 'Max','Mustermann','max@example.com'

WHERE NOT EXISTS (SELECT 1 FROM kunden);

");

    echo "<h1>PHP + Apache + MySQL läuft! Gruß Markus L.</h1>";
    echo "<p>Verbindung zur DB erfolgreich.</p>";

    $stmt = $pdo->query("SELECT id, CONCAT(vorname, ' ', nachname) AS name, email FROM
kunden ORDER BY id");

    echo "<h2>Kunden</h2><ul>";

    foreach ($stmt as $row) {

        $id = (int)$row['id'];

        $name = htmlspecialchars($row['name'] ?? '', ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');

        $email = htmlspecialchars($row['email'] ?? '', ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');

        echo "<li>#{$id} {$name} ({$email})</li>";

    }

    echo "</ul>";

} catch (PDOException $e) {

    http_response_code(500);

    echo "<h1>DB-Verbindung fehlgeschlagen</h1>";

    echo "<pre>".$e->getMessage()."</pre>";

}
```

markus@dockerserver: ~/lamp-compose/html

```
GNU nano 2.2
<?php
$dns = "mysql:host=db;dbname=firma;charset=utf8mb4";
$user = "app";
$pass = "apppass";

try {

    $pdo = new PDO($dns, $user, $pass, [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]);

    // Tabelle anlegen, falls nicht vorhanden
    $pdo->exec("

CREATE TABLE IF NOT EXISTS kunden (

    id INT AUTO_INCREMENT PRIMARY KEY,

    vorname VARCHAR(100) NOT NULL,

    nachname VARCHAR(100) NOT NULL,

    email VARCHAR(190) UNIQUE,

    erstellt_am TIMESTAMP DEFAULT CURRENT_TIMESTAMP

);

");

    // Demo-Datensatz, falls noch leer
    $pdo->exec("

INSERT INTO kunden (vorname, nachname, email)

SELECT 'Max','Mustermann','max@example.com'

WHERE NOT EXISTS (SELECT 1 FROM kunden);

");

    echo "<h1>PHP + Apache + MySQL läuft! Gruß Markus L.</h1>";
    echo "<p>Verbindung zur DB erfolgreich.</p>";

    // name aus vorname + nachname bilden
    $stmt = $pdo->query("SELECT id, CONCAT(vorname, ' ', nachname) AS name, email FROM kunden ORDER BY id");

    echo "<h2>Kunden</h2><ul>";
    foreach ($stmt as $row) {
        $id = (int)$row['id'];
        $name = htmlspecialchars($row['name'] ?? '', ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');
        $email = htmlspecialchars($row['email'] ?? '', ENT_QUOTES | ENT_SUBSTITUTE, 'UTF-8');
        echo "<li>#{$id} {$name} ({$email})</li>";
    }
    echo "</ul>";

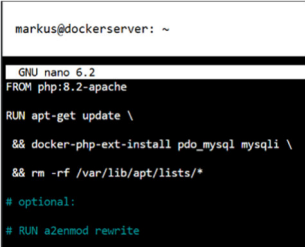
} catch (PDOException $e) {
    http_response_code(500);
    echo "<h1>DB-Verbindung fehlgeschlagen</h1>";
    echo "<pre>".$e->getMessage()."</pre>";
}
```

Dadurch war kein Neustart der Container notwendig, da der html-Ordner über ein Docker-Volume eingebunden ist.

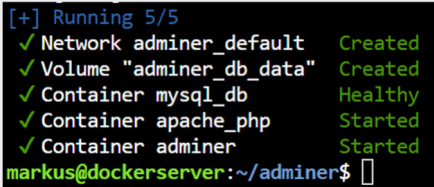
Änderungen an PHP-Dateien werden sofort im laufenden Apache/PHP-Container übernommen.

Aufruf per Browser: http://192.168.0.123:8080	
Eintrag in phpMyAdmin wurde ebenfalls erstellt →	

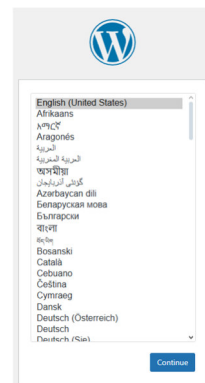
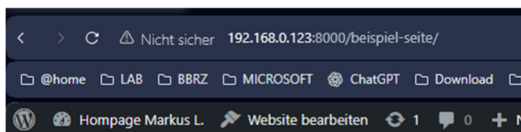
6.4 Mit Datenbank (MySQL & Adminer)

Projektordner anlegen	mkdir -p ~/adminer/{html,db}	
in den Projektordner wechseln	cd ~/adminer	
Dockerfile mit nano und Inhalt anlegen: FROM php:8.2-apache RUN apt-get update \ && docker-php-ext-install pdo_mysql mysqli \ && rm -rf /var/lib/apt/lists/* # optional: # RUN a2enmod rewrite	nano Dockerfile	

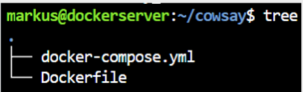
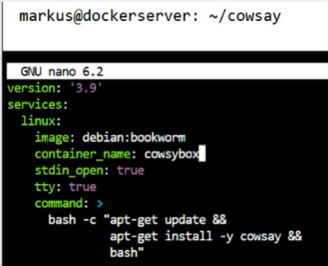
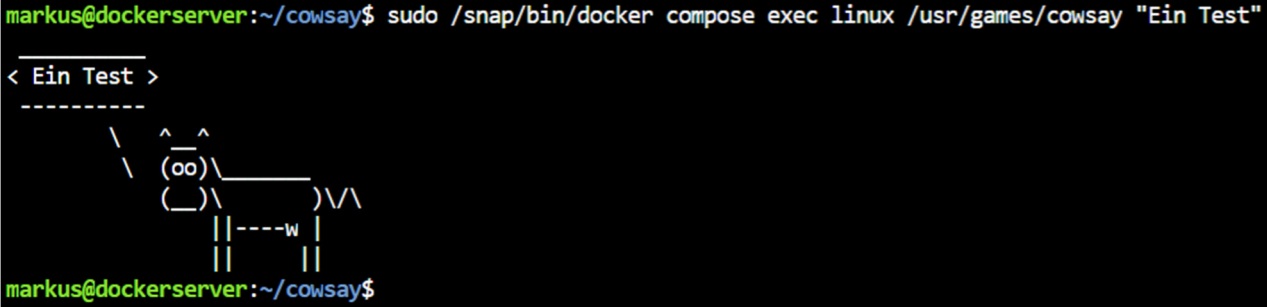
<pre>version: '3.9' services: db: image: mysql:8.0 container_name: mysql_db environment: MYSQL_ROOT_PASSWORD: root MYSQL_DATABASE: firma MYSQL_USER: app MYSQL_PASSWORD: apppass command: > --character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci volumes: - db_data:/var/lib/mysql - ./db:/docker-entrypoint-initdb.d:ro healthcheck: test: ["CMD", "mysqladmin", "ping", "-proot"] interval: 10s timeout: 5s retries: 10 restart: unless-stopped web: build: . container_name: apache_php ports: - "8080:80" volumes: - ./html:/var/www/html depends_on: db: condition: service_healthy restart: unless-stopped adminer: image: adminer container_name: adminer ports: - "8084:8080" depends_on: db: condition: service_healthy restart: unless-stopped volumes: db_data:</pre>	<pre>version: '3.9' services: db: image: mysql:8.0 container_name: mysql_db environment: MYSQL_ROOT_PASSWORD: root MYSQL_DATABASE: firma MYSQL_USER: app MYSQL_PASSWORD: apppass command: > --character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci volumes: - db_data:/var/lib/mysql - ./db:/docker-entrypoint-initdb.d:ro healthcheck: test: ["CMD", "mysqladmin", "ping", "-proot"] interval: 10s timeout: 5s retries: 10 restart: unless-stopped web: build: . container_name: apache_php ports: - "8080:80" volumes: - ./html:/var/www/html depends_on: db: condition: service_healthy restart: unless-stopped adminer: image: adminer container_name: adminer ports: - "8084:8080" depends_on: db: condition: service_healthy restart: unless-stopped volumes: db_data:</pre>
<pre><?php \$dsn = "mysql:host=db;dbname=firma;charset=utf8mb4"; \$user = "app"; \$pass = "apppass"; try { \$pdo = new PDO(\$dsn, \$user, \$pass, [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]); echo "<h1>PHP + Apache + MySQL läuft! Gruß Markus L.</h1>"; echo "<p>Verbindung zur DB erfolgreich.</p>"; // Demo-Tabelle sicherstellen \$pdo->query("CREATE TABLE IF NOT EXISTS kunden (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(120) NOT NULL)"); \$count = (int)\$pdo->query("SELECT COUNT(*) FROM kunden")->fetchColumn(); echo "<p>Datensätze in 'kunden': \$count</p>"; } catch (PDOException \$e) { http_response_code(500); echo "<h1>DB-Verbindung fehlgeschlagen</h1>"; echo "<pre>".htmlspecialchars(\$e->getMessage())."</pre>"; }</pre>	<pre><?php \$dsn = "mysql:host=db;dbname=firma;charset=utf8mb4"; \$user = "app"; \$pass = "apppass"; try { \$pdo = new PDO(\$dsn, \$user, \$pass, [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]); echo "<h1>PHP + Apache + MySQL läuft! Gruß Markus L.</h1>"; echo "<p>Verbindung zur DB erfolgreich.</p>"; // Demo-Tabelle sicherstellen \$pdo->query("CREATE TABLE IF NOT EXISTS kunden (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(120) NOT NULL)"); \$count = (int)\$pdo->query("SELECT COUNT(*) FROM kunden")->fetchColumn(); echo "<p>Datensätze in 'kunden': \$count</p>"; } catch (PDOException \$e) { http_response_code(500); echo "<h1>DB-Verbindung fehlgeschlagen</h1>"; echo "<pre>".htmlspecialchars(\$e->getMessage())."</pre>"; }</pre>
Initial-Daten db/init.sql (sicherheitshalber) <pre>CREATE TABLE IF NOT EXISTS kunden (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(120) NOT NULL); INSERT INTO kunden (name, email) VALUES ('Max Mustermann', 'max@example.com'), ('Erika Musterfrau', 'erika@example.com');</pre>	<pre>CREATE TABLE IF NOT EXISTS kunden (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100) NOT NULL, email VARCHAR(120) NOT NULL); INSERT INTO kunden (name, email) VALUES ('Max Mustermann', 'max@example.com'), ('Erika Musterfrau', 'erika@example.com');</pre>

 Erkennen: Ist Docker über Snap oder apt installiert? → /snap/bin/docker	Wenn <i>which docker</i> → /snap/bin/docker, dann nutzt du Snap. Wenn <i>which docker</i> → /usr/bin/docker, dann apt/Offiziell.
Shell-Befehl ausführen Mit <code>export PATH="\$PATH:/snap/bin"</code> sagst du deiner Shell, wo sie nach Programmen suchen soll. Ohne diesen Eintrag findet die Shell docker (aus dem Snap) nicht → „command not found“.	<code>export PATH="\$PATH:/snap/bin"</code>
in das Verzeichnis wechseln und die Befehle nacheinander ausführen...	<pre>cd ~/adminer sudo docker compose build --no-cache sudo docker compose up -d</pre> 
Zugriff auf Web (PHP/Apache): http://192.168.0.123:8080	
Zugriff auf Adminer: http://192.168.0.123:8084	

6.5 Wordpress & MySQL

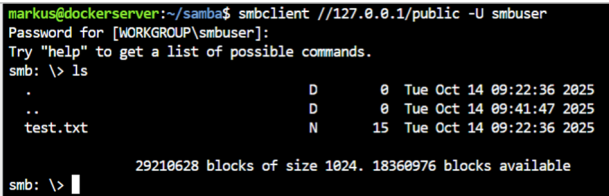
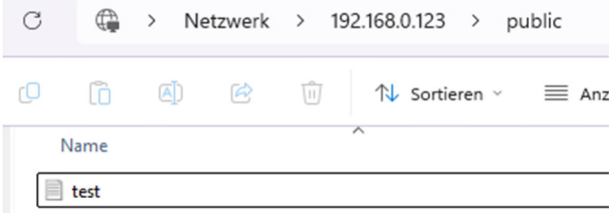
Projektordner anlegen	<code>mkdir -p ~/wordpress</code>
in den Projektordner wechseln	<code>cd ~/wordpress</code>
nano docker-compose.yml <pre> version: "3.9" services: db: image: mysql:8.0.27 container_name: wp_db command: --default-authentication-plugin=mysql_native_password environment: MYSQL_ROOT_PASSWORD: somewordpress MYSQL_DATABASE: wordpress MYSQL_USER: wordpress MYSQL_PASSWORD: wordpress volumes: - db_data:/var/lib/mysql healthcheck: test: ["CMD", "mysqladmin", "ping", "-psomewordpress"] interval: 10s timeout: 5s retries: 10 restart: unless-stopped wordpress: image: wordpress:php8.2-apache # explizit, stabil container_name: wordpress ports: - "8000:80" # Browser -> Container environment: WORDPRESS_DB_HOST: db:3306 WORDPRESS_DB_USER: wordpress WORDPRESS_DB_PASSWORD: wordpress WORDPRESS_DB_NAME: wordpress volumes: - wp_data:/var/www/html # persistente WP-Dateien/Uploads depends_on: db: condition: service_healthy restart: unless-stopped volumes: db_data: wp_data: </pre>	<pre> 1 version: "3.9" 2 3 services: 4 db: 5 image: mysql:8.0.25 6 container_name: wp_db 7 command: --default-authentication-plugin=mysql_native_password 8 environment: 9 MYSQL_ROOT_PASSWORD: somewordpress 10 MYSQL_DATABASE: wordpress 11 MYSQL_USER: wordpress 12 MYSQL_PASSWORD: wordpress 13 volumes: 14 - db_data:/var/lib/mysql 15 healthcheck: 16 test: ["CMD", "mysqladmin", "ping", "-psomewordpress"] 17 interval: 10s 18 timeout: 5s 19 retries: 10 20 restart: unless-stopped 21 22 wordpress: 23 image: wordpress:php8.2-apache # explizit, stabil 24 container_name: wordpress 25 ports: 26 - "8000:80" # Browser -> Container 27 environment: 28 WORDPRESS_DB_HOST: db:3306 29 WORDPRESS_DB_USER: wordpress 30 WORDPRESS_DB_PASSWORD: wordpress 31 WORDPRESS_DB_NAME: wordpress 32 volumes: 33 - wp_data:/var/www/html # persistente WP-Dateien/Uploads 34 depends_on: 35 db: 36 condition: service_healthy 37 restart: unless-stopped 38 39 volumes: 40 db_data: 41 wp_data: 42 </pre>
Falls Docker per Snap installiert ist	<code>export PATH="\$PATH:/snap/bin"</code>
Starten	<code>sudo docker compose up -d</code>
Stoppen	<code>sudo docker-compose down</code>
Aufruf im Browser: http://192.168.0.123:8000 → WordPress-Setup durchklicken (Sprache, Titel, Admin-User anlegen).	
 Homepage Markus L.	

6.6 Debian & Cowsay

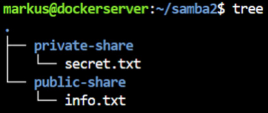
Erklärung	Beispiel
Projektordner anlegen	<code>mkdir -p ~/cowsay</code> 
in den Projektordner wechseln	<code>cd ~/cowsay</code>
Dockerfile erstellen (Debian mit cowsay vorinstalliert) <code>sudo nano Dockerfile</code>	Inhalt FROM debian:bookworm-slim ENV DEBIAN_FRONTEND=noninteractive RUN apt-get update \ && apt-get install -y --no-install-recommends cowsay fortune-mod \ && rm -rf /var/lib/apt/lists/* # cowsay liegt unter /usr/games/cowsay
Eintrag in der docker-compose.yml <pre>services: cowsay: build: . container_name: cowsybox tty: true command: ["sleep", "infinity"]</pre>	<code>sudo nano docker-compose.yml</code> 
Starten	<code>sudo docker compose up -d --build</code>
zeigt den Status der Dienste aus der aktuellen Compose-Umgebung (dem Ordner mit deiner docker-compose.yml).	<code>docker compose ps</code>
Stoppen	<code>sudo docker-compose down</code>
Installiert cowsay im laufenden Container (Service linux).	<code>sudo /snap/bin/docker compose exec linux bash -lc 'apt-get update && apt-get install -y cowsay'</code>
Führt cowsay im Container aus und gibt den Spruch „Ein Test“ zurück.	<code>sudo /snap/bin/docker compose exec linux /usr/games/cowsay "Ein Test"</code> 

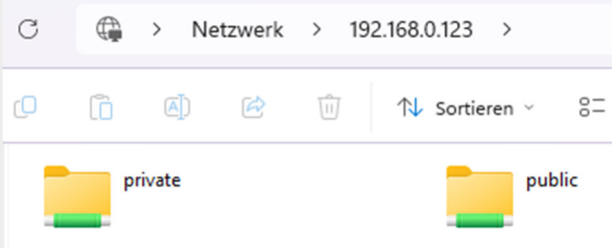
6.7 Alpine & Samba(einfach)

Erklärung	Beispiel
Projektordner anlegen	<code>mkdir -p ~/samba/shared</code>
in den Projektordner wechseln	<code>cd ~/cowsay</code>
erstellen einer Textdatei mit dem Eintrag unter „shared“	<code>echo "eh nur bla bla" > ~/samba/shared/test.txt</code>

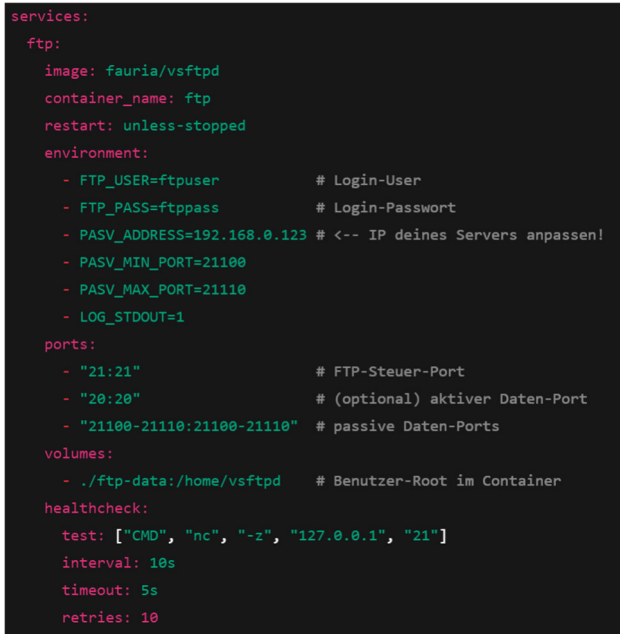
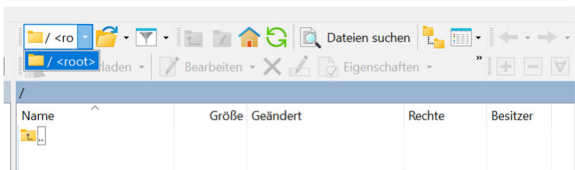
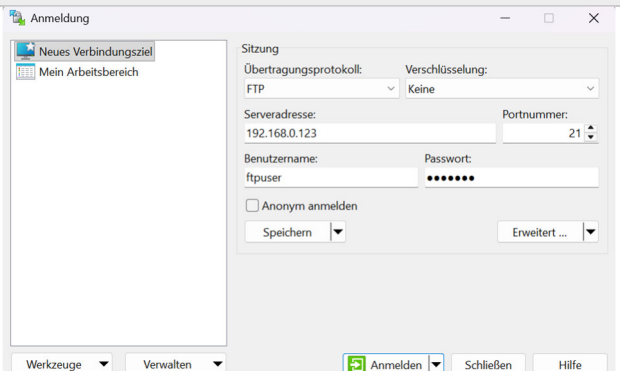
Wechsel in den Ordner „samba“	<code>cd ~/samba</code>
Mit Hilfe von nano die .yml Datei erstellen services: samba: image: dperson/samba container_name: samba restart: unless-stopped ports: - "139:139" - "445:445" command: > -u "smbuser;smbpass" -s "public;/mount;yes;no;no;smbuser" -w "WORKGROUP" volumes: - ./shared:/mount	<code>sudo nano docker-compose.yml</code> markus@dockerserver: ~/samba 
Samba-Client unter Linux installieren	<code>sudo apt -y install smbclient</code>
Starten	<code>sudo /snap/bin/docker compose -f ~/samba/docker-compose.yml up -d</code>
stoppen & entfernen	<code>sudo /snap/bin/docker compose -f ~/samba/docker-compose.yml down</code>
Zugriff von Linux <code>smbclient //127.0.0.1/public -U smbuser</code>	
Zugriff von Windows-Explorer <code>\\192.168.0.123\public</code>	

6.8 Samba erweitert

Erklärung	Beispiel
Projektordner anlegen 	<code>mkdir -p ~/samba2/public-share ~/samba2/private-share</code>

Mit Hilfe von nano die docker-compose Datei erstellen <pre> services: samba: image: dperson/samba container_name: samba restart: unless-stopped ports: - "139:139" - "445:445" command: > -u "smbuser;smbpass" -s "public:/mount/public;yes;no;yes" -s "private:/mount/private;yes;no;no;smbuser" -w "WORKGROUP" volumes: - ./public-share:/mount/public - ./private-share:/mount/private </pre>	<pre> markus@dockerserver: ~/samba2 GNU nano 6.2 services: samba: image: dperson/samba container_name: samba restart: unless-stopped ports: - "139:139" - "445:445" # Benutzer + Shares über command (robust gemäß Image-Doku) command: > -u "smbuser;smbpass" -s "public:/mount/public;yes;no;yes" -s "private:/mount/private;yes;no;no;smbuser" -w "WORKGROUP" volumes: - ./public-share:/mount/public - ./private-share:/mount/private </pre>
erstellt die Datei „info.txt“ mit dem Inhalt „Öffentliche Datei“	<pre>echo "Öffentliche Datei" > ~/samba2/public-share/info.txt</pre>
erstellt die Datei „info.txt“ mit dem Inhalt „Private Datei“	<pre>echo "Private Datei" > ~/samba2/private-share/secret.txt</pre>
Wechselt in den Ordner „samba2“	<pre>cd ~/samba2</pre>
Einmal pro Session (falls docker nicht gefunden wird)	<pre>export PATH="\$PATH:/snap/bin"</pre>
Start	<pre>sudo docker-compose up -d</pre>
Stopp	<pre>sudo docker-compose down</pre>
beide Ordner befinden sich in der WIN-Freigabe auf beide kann ich zugreifen	
Zugriff unter Linux auf localhost/public smbclient //127.0.0.1/public -U smbuser	<pre> markus@dockerserver:~/samba2\$ smbclient //127.0.0.1/public -U smbuser Password for [WORKGROUP\smbuser]: Try "help" to get a list of possible commands. smb: \> ls . D 0 Tue Oct 14 11:21:18 2025 .. D 0 Tue Oct 14 11:57:12 2025 info.txt N 19 Tue Oct 14 11:21:18 2025 29210628 blocks of size 1024. 18358644 blocks available smb: \> </pre>
Zugriff unter Linux auf localhost/private smbclient //127.0.0.1/private -U smbuser	<pre> markus@dockerserver:~/samba2\$ smbclient //127.0.0.1/private -U smbuser Password for [WORKGROUP\smbuser]: Try "help" to get a list of possible commands. smb: \> ls . D 0 Tue Oct 14 11:22:42 2025 .. D 0 Tue Oct 14 11:57:12 2025 secret.txt N 14 Tue Oct 14 11:22:42 2025 29210628 blocks of size 1024. 18358644 blocks available smb: \> </pre>

6.9 FTP-Server

Erklärung	Beispiel
Erstellung des Ordners „ftp-data“ unter „ftp“	<code>mkdir -p ~/ftp/ftp-data</code>
erstellt die Datei „info.txt“ mit dem Inhalt „FTP-Testdatei“	<code>echo "FTP-Testdatei" > ~/ftp/ftp-data/info.txt</code>
Wechselt in den Ordner „ftp“	<code>cd ~/ftp</code>
Mit Hilfe von nano die docker-compose Datei erstellen <pre> services: ftp: image: fauria/vsftpd container_name: ftp restart: unless-stopped environment: - FTP_USER=ftpuser # Login-User - FTP_PASS=ftppass # Login-Passwort - PASV_ADDRESS=192.168.0.123 # <-- IP deines Servers anpassen! - PASV_MIN_PORT=21100 - PASV_MAX_PORT=21110 - LOG_STDOUT=1 ports: - "21:21" # FTP-Steuer-Port - "20:20" # (optional) aktiver Daten-Port - "21100-21110:21100-21110" # passive Daten-Ports volumes: - ./ftp-data:/home/vsftpd # Benutzer-Root im Container healthcheck: test: ["CMD", "nc", "-z", "127.0.0.1", "21"] interval: 10s timeout: 5s retries: 10 </pre>	
Starten	<code>sudo /snap/bin/docker compose up -d</code>
Stoppen & löschen	<code>sudo /snap/bin/docker compose down</code>
Zugriff unter Linux	
FTP-Client installieren unter Linux (CLI)	<code>sudo apt update && sudo apt install -y ftp</code>
Zugriff testen	<pre> markus@dockerserver:~/ftp\$ ftp 192.168.0.123 Connected to 192.168.0.123. 220 (vsFTPd 3.0.2) Name (192.168.0.123:markus): ftpuser 331 Please specify the password. Password: 230 Login successful. </pre>
ftp 192.168.0.123	
Zugriff unter Windows	
Beliebigen FTP-Client nutzen → WinSCP neues Verbindungsziel anlegen. 	
und ich bin verbunden!!!	