35_Docker Compose mit Ubuntu Server

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

Inhalt

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)

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

2.2 User & Zugangsdaten (Software)

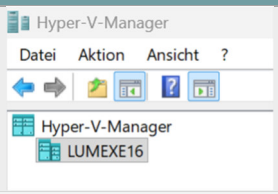
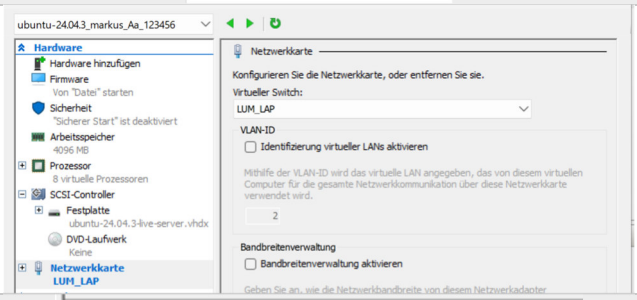
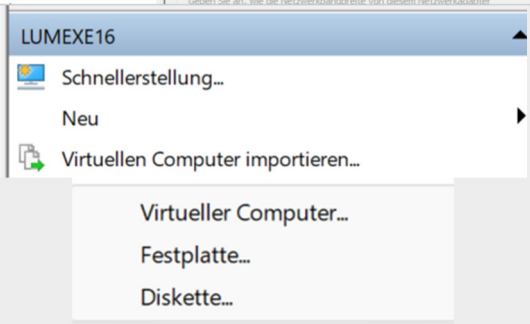
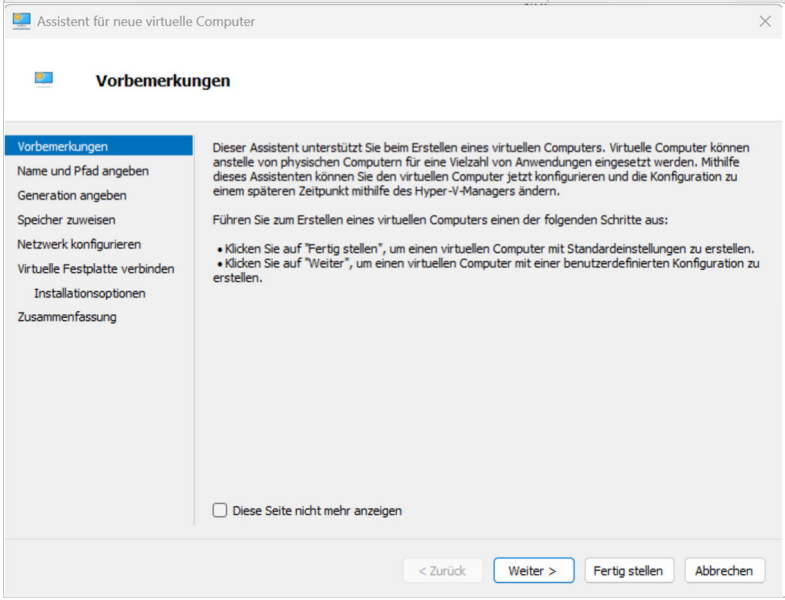
	Geräte #	Anwendung	Benutzername	Passwort	Gerätename	domain (AD)
Client						

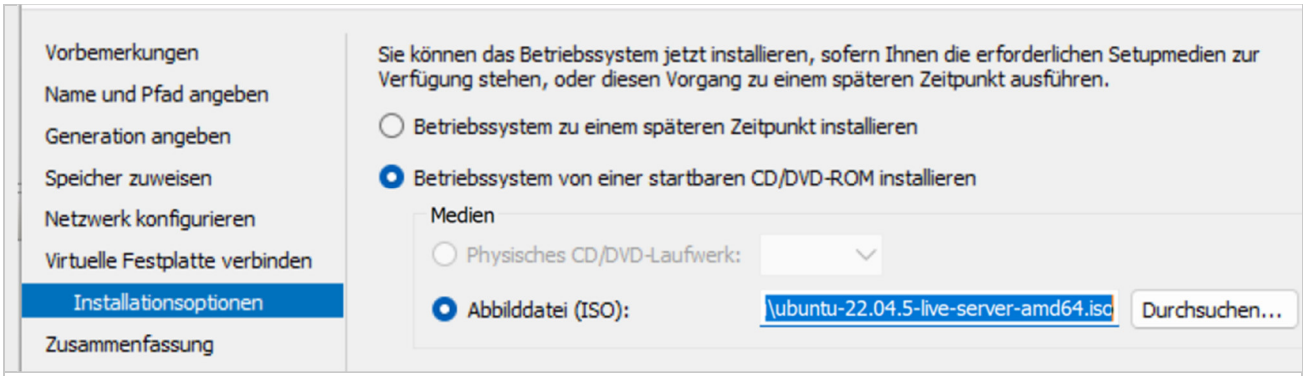
	Geräte #	Anwendung	Benutzername	Passwort	Hostname
Server					

3. Installation einer VM (Hyper-V)

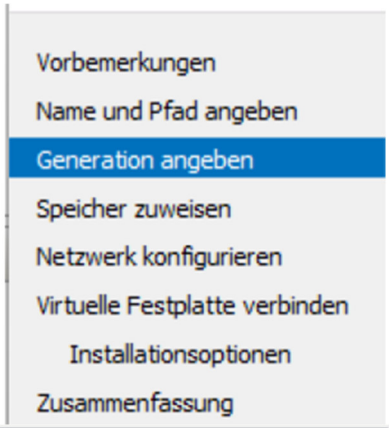
Installation eines UBUNTU-SERVERS von https://releases.ubuntu.com/jammy

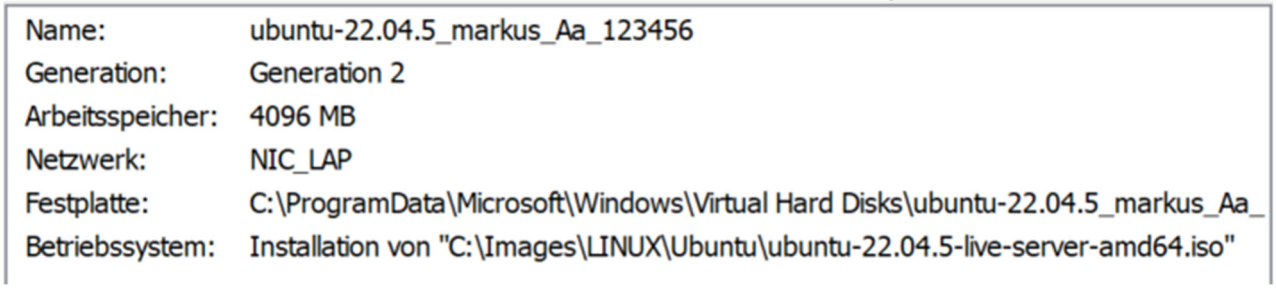
Der download liegt als ISO-Datei vor. ubuntu-22.04.5-live-server-amd64

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.



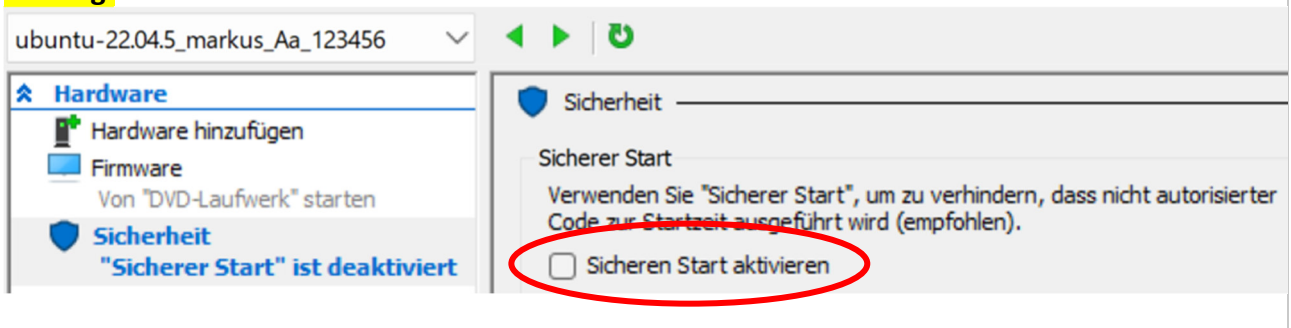
Name und Pfad angeben**: ubuntu-22.04.5_markus_Aa_123456**
Generation angeben: **2.Generation**
Speicher zuweisen: 4096 MB
Netzwerk konfigurieren: NIC_LAP
Virtuelle Festplatte verbinden: virtuelle Festplatte erstellen
Größe: 60GB
Installationsoptinen: Betriebssystem von einem startbaren Medium installieren →Abbilddatei (ISO) → Speicherort der ISO-Datei!!!
Zusammenfassung: → Fertigstellen



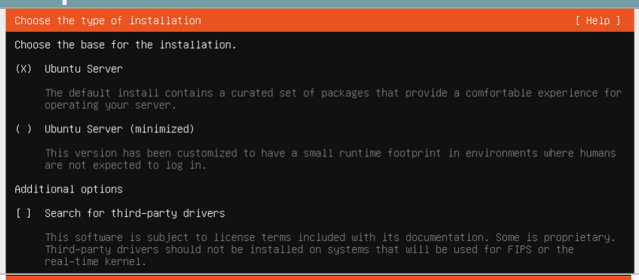
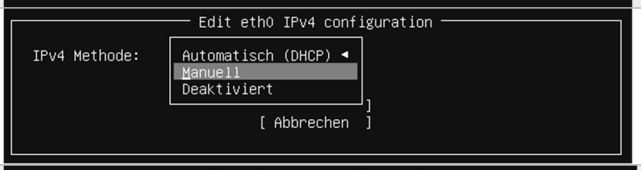
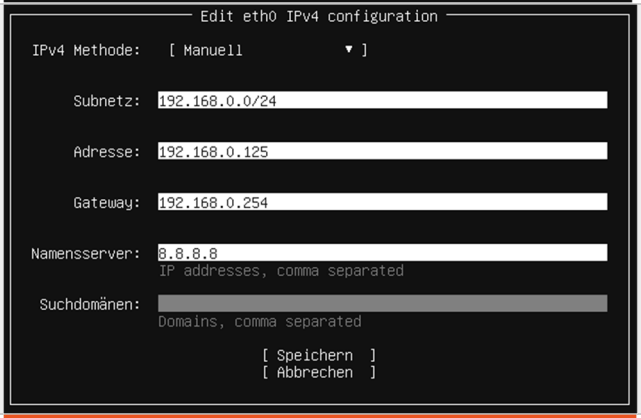
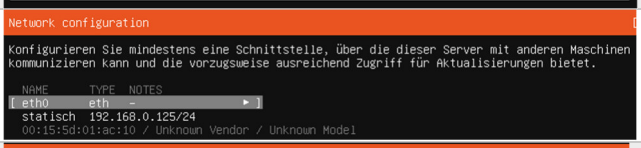


Die VM wird erstellt und erscheint in der Liste der VM´s!

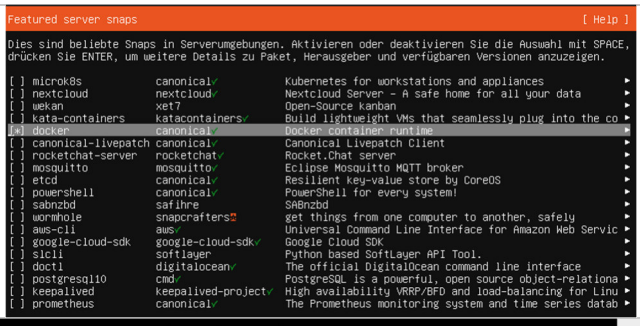
Wichtig:



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	

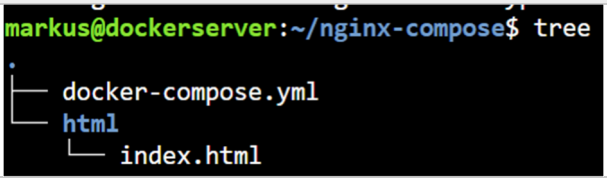
Wähle nur **docker** aus und drücke Enter
Damit bekommst du das offizielle Docker Snap-Paket, das sofort lauffähig ist,
und du kannst danach direkt mit **docker compose oder docker-compose** weitermachen!



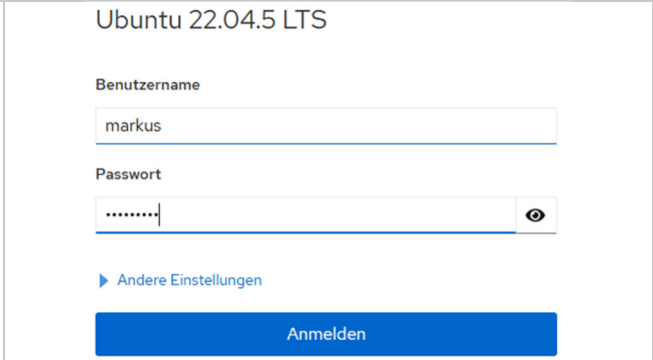
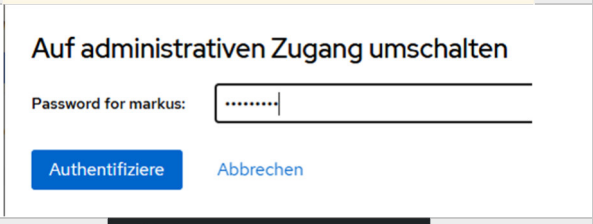
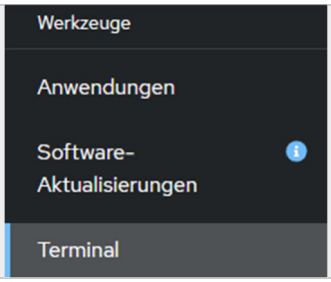

```
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-157-generic x86_64)

 * 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 →	sudo apt update sudo apt install tree
mit dem Befehl „**tree**“ wird die aktuelle Ordnerstruktur angezeigt! später sehr hilfreich!	

5.2 Installation cockpit

Erklärung	Beispiel
Cockpit installieren	sudo apt install cokpit
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	sudo docker –version
	`markus@dockerserver:~$ sudo docker --version`
	`Docker version 28.1.1+1, build 068a01e`

docker compose version prüfen	sudo docker compose --version
	`markus@dockerserver:~$ sudo docker compose version`
	`Docker Compose version v2.33.1`

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:
	`markus@dockerserver:~$ sudo groupadd docker`
	`sudo usermod -aG docker $USER`
	`newgrp dockermarkus@dockerserver:~$`
	`markus@dockerserver:~$`

Test	sudo docker ps
	`markus@dockerserver:~$ sudo docker ps`
	`CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES`
	`markus@dockerserver:~$`

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:

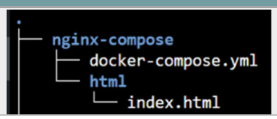
Für neue Projekte solltest du immer die moderne Syntax docker compose (ohne Bindestrich) verwenden.

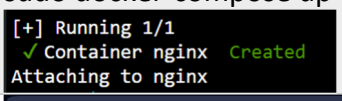
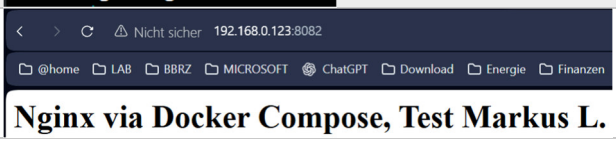
5.5 Docker Kommandos/Befehle

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

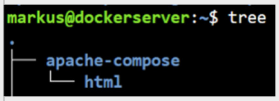
6. Beispiele mit „docker“

6.1 Einfacher Webserver mit Nginx

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

mit nano eine .yml Datei erstellen und folgenden Eintrag darin machen: **sudo nano docker-compose.yml** 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	
container starten →	sudo docker-compose up -d 
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.	 mkdir -p ~/apache-compose/html
erstellt den Eintrag „Apache läuft in Docker & Startseite von Markus“ in der Datei index.html	echo '<!doctype html><html><head><meta charset="utf-8"><title>Apache </title></head><body> Apache läuft in Docker Startseite von Markus </body></html>' > ~/apache-compose/html/index.html
in den Ordner „apache-compose“ wechseln	cd ~/apache-compose
nano-Editor öffnen und folgenden Eintrag hinzufügen: 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	sudo nano docker-compose.yml 
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	mkdir -p ~/lamp-compose/html ~/lamp-compose/db
	`markus@dockerserver:~/lamp-compose$ tree`
	`.`
	`├── db`
	`└── html`

wechselt in den Ordner „lamp-compose/html“	cd ~/lamp-compose/html
eine Datei mit dem Namen „index.php“ erstellen.	sudo touch index.php

Eintrag in der index.php Datei:
`<?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 = $pdo->query("SELECT id, name, email FROM kunden ORDER BY id");`
`echo "<h2>Kunden</h2><ul>";`
`foreach ($stmt as $row) {`
`echo "<li>#{$row['id']} {$row['name']} ({$row['email']})</li>";`
`}`
`echo "</ul>";`
`} catch (PDOException $e) {`
`http_response_code(500);`
`echo "<h1>DB-Verbindung fehlgeschlagen</h1>";`
`echo "<pre>".$e->getMessage()."</pre>";`
`}`
	sudo nano index.php
	`GNU nano 6.2`
	`<?php`
	`$dsn = "mysql:host=db;dbname=firma;charset=utf8mb4";`
	`$user = "app";`
	`$pass = "apppass";`
	`try {`
	`$pdo = new PDO($dsn, $user, $pass, [`
	`PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION`
	`]);`
	`echo "<h1>PHP + Apache + MySQL läuft 🎉</h1>";`
	`echo "<p>Verbindung zur DB erfolgreich.</p>";`
	`$stmt = $pdo->query("SELECT id, name, email FROM kunden ORDER BY id");`
	`echo "<h2>Kunden</h2><ul>";`
	`foreach ($stmt as $row) {`
	`echo "<li>#{$row['id']} {$row['name']} ({$row['email']})</li>";`
	`}`
	`echo "</ul>";`
	`} catch (PDOException $e) {`
	`http_response_code(500);`
	`echo "<h1>DB-Verbindung fehlgeschlagen</h1>";`
	`echo "<pre>".$e->getMessage()."</pre>";`
	`}`

in das Verezichnis „lamp-compose“ wechseln und Eintrag in der docker-compose.yml:	cd lamp-compose/ sudo nano docker-compose.yml

```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:
```

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.**

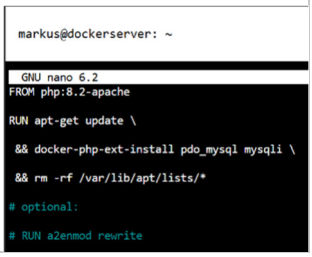
```
<?php
$dsn  = "mysql:host=db;dbname=firma;charset=utf8mb4";

$user = "app";

$pass = "apppass";

try {

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

 $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

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)

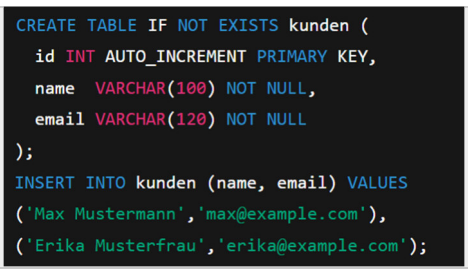
Projektordner anlegen **in den Projektordner wechseln**	mkdir -p ~/adminer/{html,db} cd ~/adminer
Dockerfile mit nano und Inhalt anlegen: FROM php:8.2-apache RUN apt-get update \ && docker-php-ext-install pdo_mysql mysqli \ && rm -rf /var/lib/apt/lists/* # optional: # RUN a2enmod rewrite	nano Dockerfile 

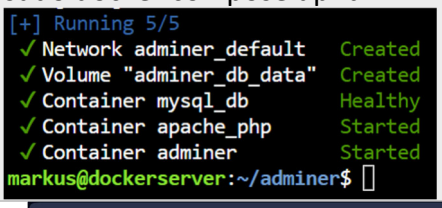
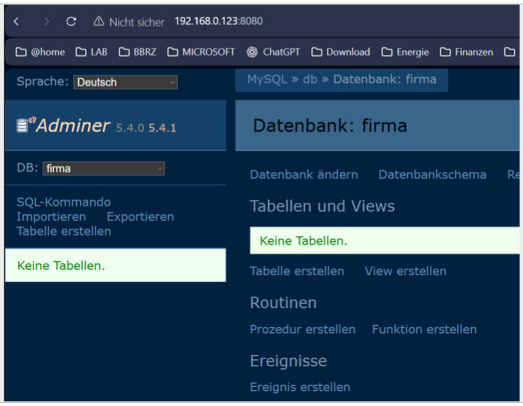
```yaml
version: '3.9'

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:
```

```php
<?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>";
}
```

Initial-Daten db/init.sql (sicherheitshalber)

```sql
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');
```



Erkennen: Ist Docker über Snap oder apt installiert? ➔ **/snap/bin/docker**	Wenn *which docker* → /snap/bin/docker, dann nutzt du Snap. Wenn which docker → /usr/bin/docker, dann apt/Offiziell.
Shell-Befehl ausführen Mit export PATH="$PATH:/snap/bin" sagst du deiner Shell, wo sie nach Programmen suchen soll. Ohne diesen Eintrag findet die Shell docker (aus dem Snap) nicht → „command not found“.	export PATH="$PATH:/snap/bin"
in das Verzeichnis wechseln und die Befehle nacheinander ausführen…	cd ~/adminer sudo docker compose build --no-cache sudo docker compose up -d 
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

mkdir -p ~/wordpress

in den Projektordner wechseln

cd ~/wordpress

nano docker-compose.yml

```
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:
```

```yaml
version: "3.9"

services:
  db:
    image: mysql:8.0.25
    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:
```

Falls Docker per Snap installiert ist

export PATH="$PATH:/snap/bin"

Starten

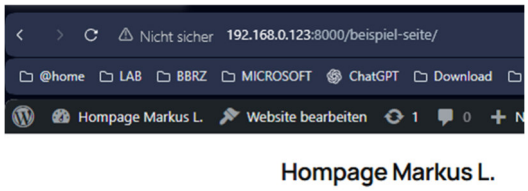
sudo docker compose up -d

Stoppen

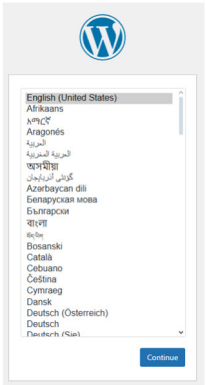
sudo docker-compose down

Aufruf im Browser:
http://192.168.0.123:8000

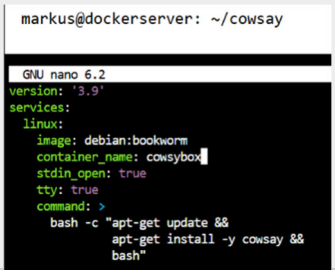
→ WordPress-Setup durchklicken (Sprache, Titel, Admin-User anlegen).



Hompage Markus L.



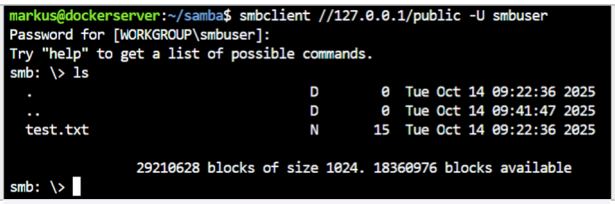
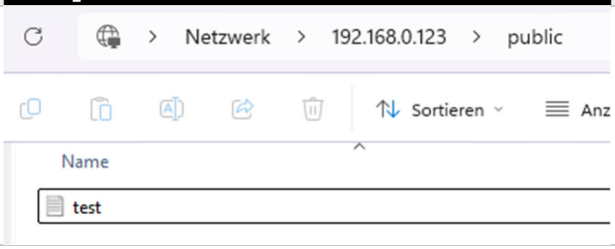
6.6 Debian & Cowsay

Erklärung	Beispiel
Projektordner anlegen **in den Projektordner wechseln**	mkdir -p ~/cowsay cd ~/cowsay 
Dockerfile erstellen (Debian mit cowsay vorinstalliert) sudo nano Dockerfile	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 **services:** **cowsay:** **build: .** **container_name: cowsybox** **tty: true** **command: ["sleep","infinity"]**	sudo nano docker-compose.yml 
Starten	sudo docker compose up -d --build
zeigt den Status der Dienste aus der aktuellen Compose-Umgebung (dem Ordner mit deiner docker-compose.yml).	docker compose ps
Stoppen	sudo docker-compose down
Installiert cowsay im laufenden Container (Service linux).	sudo /snap/bin/docker compose exec linux bash -lc 'apt-get update && apt-get install -y cowsay'
Führt cowsay im Container aus und gibt den Spruch „Ein Test“ zurück.	sudo /snap/bin/docker compose exec linux /usr/games/cowsay "Ein Test"

```
markus@dockerserver:~/cowsay$ sudo /snap/bin/docker compose exec linux /usr/games/cowsay "Ein Test"
 __________
< Ein Test >
 ----------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
markus@dockerserver:~/cowsay$
```

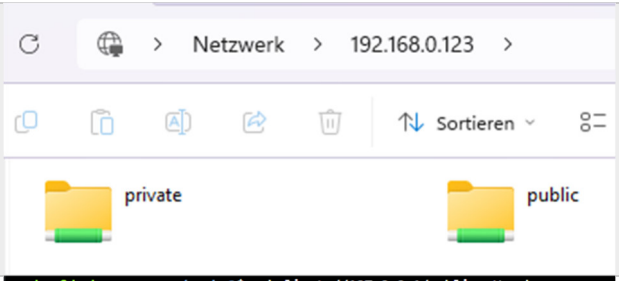
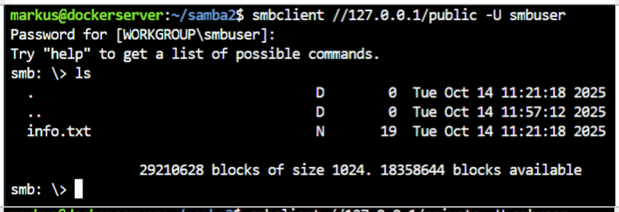
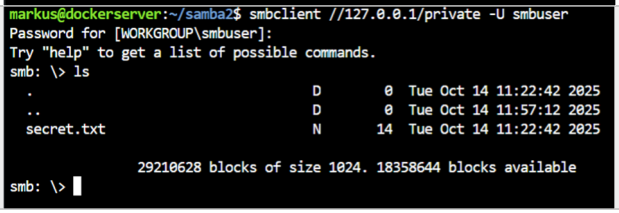
6.7 Alpine & Samba(einfach)

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

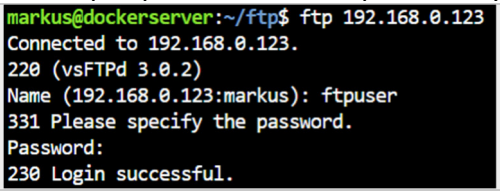
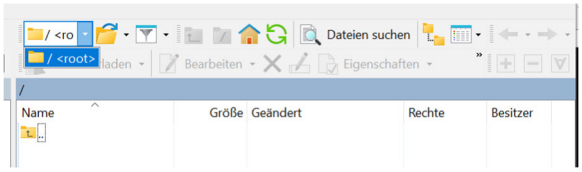
Wechsel in den Ordner „samba“	cd ~/samba
Mit Hilfe von nano die .yml Datei erstellen **services:** **samba:** **image: dperson/samba** **container_name: samba** **restart: unless-stopped** **ports:** **- "139:139"** **- "445:445"** **command: >** **-u "smbuser;smbpass"** **-s "public;/mount;yes;no;no;smbuser"**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

6.8 Samba erweitert

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

Mit Hilfe von nano die docker-compose 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/public;yes;no;yes" -s "private;/mount/private;yes;no;no;smbuser" -w "WORKGROUP" volumes: - ./public-share:/mount/public - ./private-share:/mount/private	
erstellt die Datei „info.txt“ mit dem Inhalt „Öffentliche Datei“	echo "Öffentliche Datei" > ~/samba2/public-share/info.txt
erstellt die Datei „info.txt“ mit dem Inhalt „Private Datei“	echo "Private Datei" > ~/samba2/private-share/secret.txt
Wechselt in den Ordner „samba2“	cd ~/samba2
Einmal pro Session (falls docker nicht gefunden wird)	export PATH="$PATH:/snap/bin"
Start	sudo docker-compose up -d
Stopp	sudo docker-compose down
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**	
Zugriff unter Linux auf localhost/public **smbclient //127.0.0.1/private -U smbuser**	

6.9 FTP-Server

Erklärung	Beispiel
Erstellung des Ordners „ftp-data“ unter „ftp“	mkdir -p ~/ftp/ftp-data
erstellt die Datei „info.txt“ mit dem Inhalt „FTP-Testdatei“	echo "FTP-Testdatei" > ~/ftp/ftp-data/info.txt
Wechselt in den Ordner „ftp“	cd ~/ftp
Mit Hilfe von nano die docker-compose Datei erstellen **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**	
Starten	sudo /snap/bin/docker compose up -d
Stoppen & löschen	sudo /snap/bin/docker compose down
Zugriff unter Linux	
FTP-Client installieren unter **Linux (CLI)**	sudo apt update && sudo apt install -y ftp
Zugriff testen **ftp 192.168.0.123**	
Zugriff unter Windows	
Beliebigen FTP-Client nutzen → WinSCP neues Verbindungsziel anlegen.  und ich bin verbunden!!!