



# **Linux Webservices Milestone 1**

Baisangur Dudayev  
r0930201

2CCS01

**CAMPUS**

**Geel**



**Academic year 2023-2024**



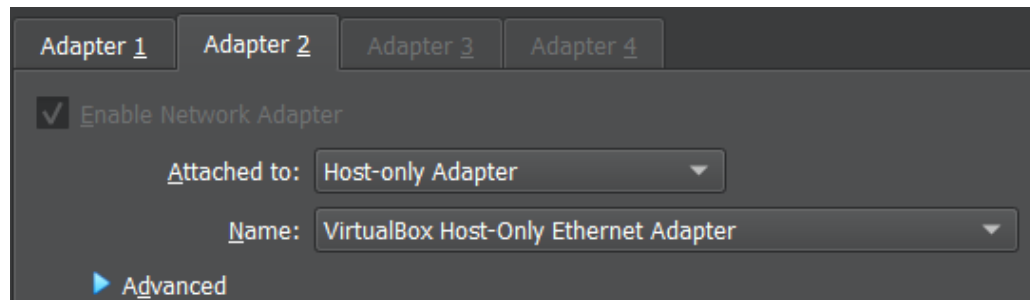
## Inhoudstafel

Inhoudstafel .....	3
Stappenplan .....	4
Ip & netwerk kaart toevoegen aan virtual machine.....	4
Installatie, opstelling & werking in virtual machine .....	5
Link naar video .....	7
Code en commentaar .....	8
docker-compose.yaml.....	8
Dockerfile .....	12
index.php.....	13
init.sql	15
default.conf.....	15
Reflectie AI .....	17
Bronnen AI (chat-GPT) .....	18
Prompt 1 help with mysql, index.php.....	18
Prompt 2 print table values .....	19
Prompt 3 create init.sql.....	20
Prompt 4 non interactive installation .....	22
Prompt 5 set containername.....	23
Prompt 6, 7, 8 load balancing with nginx .....	24
Bronnen.....	28

# Stappenplan

## Ip & netwerk kaart toevoegen aan virtual machine

Om te beginnen heb ik een ubuntu virtual machine aangemaakt in virtualbox met de image "ubuntu-22.04.3-desktop-amd64.iso". Na de installatie heb ik een host only adapter toegevoegd in virtual box en een ip toegevoegd via netplan.



```
lshw -class network
```

Command gebruikt om de logische naam van mijn nieuwe netwerk kaart te zien.

```
baisangur@user-VirtualBox:~/milestone1$ lshw -class network
WARNING: you should run this program as super-user.
*-network:0
  description: Ethernet interface
  product: 82540EM Gigabit Ethernet Controller
  vendor: Intel Corporation
  physical id: 3
  bus info: pci@0000:00:03.0
  logical name: enp0s3
  version: 02
  serial: 08:00:27:3a:30:e5
  size: 1Gbit/s
  capacity: 1Gbit/s
  width: 32 bits
  clock: 66MHz
  capabilities: bus_master cap_list ethernet physical tp 10bt 10bt-fd 100bt 100bt-fd 1000bt-fd autonegotia
tion
  configuration: autonegotiation=on broadcast=yes driver=e1000 driverversion=6.2.0-35-generic duplex=full
ip=10.0.2.15 latency=64 link=yes mngnt=255 multicast=yes port=twisted pair speed=1Gbit/s
  resources: irq:19 memory:f0200000-f021ffff ioport:d020(size=8)
*-network:1
  description: Ethernet interface
  product: 82540EM Gigabit Ethernet Controller
  vendor: Intel Corporation
  physical id: 8
  bus info: pci@0000:00:08.0
  logical name: enp0s8
  version: 02
  serial: 08:00:27:9e:5a:c6
  size: 1Gbit/s
  capacity: 1Gbit/s
  width: 32 bits
  clock: 66MHz
  capabilities: bus_master cap_list ethernet physical tp 10bt 10bt-fd 100bt 100bt-fd 1000bt-fd autonegotia
tion
  configuration: autonegotiation=on broadcast=yes driver=e1000 driverversion=6.2.0-35-generic duplex=full
ip=192.168.56.44 latency=64 link=yes mngnt=255 multicast=yes port=twisted pair speed=1Gbit/s
  resources: irq:16 memory:f0820000-f083ffff ioport:d240(size=8)
WARNING: output may be incomplete or inaccurate, you should run this program as super-user.
```

Met nano heb ik de ip address 192.168.56.44/24 toegevoegd in de netplan file "01-network-manager-all.yaml"

```
baisangur@user-VirtualBox:~/milestone1$ cat /etc/netplan/01-network-manager-all.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s8:
      addresses:
        - 192.168.56.44/24
```

```
sudo netplan apply
```

heb ik gebruikt om de wijziging effectief te maken.

```
baisangur@user-VirtualBox:~/milestone1$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:3a:30:e5 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 85791sec preferred_lft 85791sec
    inet6 fe80::f642:964f:6191:b0a5/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:9e:5a:c6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.44/24 brd 192.168.56.255 scope global noprefixroute enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe9e:5ac6/64 scope link
        valid_lft forever preferred_lft forever
```

## Installatie, opstelling & werking in virtual machine

```
wget -O docker.sh https://get.docker.com/
bash docker.sh

sudo usermod -aG docker Baisangur
newgrp docker

docker login
```

dit zijn de commandos die voorkomen in de Thomas More cursus van Linuw Webservices. We halen de docker script van het internet en voeren het uit. Daarna geven we nieuwe rechten aan de gebruiker en voegen we het toe aan de groep die we daarna maken. Na docker login gaf ik mijn naam en password op voor mijn docker account en ben ik ingelogd.

de folder milestone1 bevat al mijn bestanden die ik heb aangemaakt voor de project.

```
baisangur@user-VirtualBox:~/milestone1$ ls
default.conf  docker-compose.yaml  Dockerfile  index.php  init.sql
```

```
docker compose up -d --build
```

Dit commando gebruik ik om mijn containers op te starten. -d zodat ze in de achtergrond draaien en --build om mijn custom image opnieuw te bouwen als het niet up to date is.

```

baisangur@user-VirtualBox:~/milestone1$ docker compose up -d --build
[+] Building 2.0s (16/18)
=> [apache3 internal] load build definition from Dockerfile
=> => transferring dockerfile: 581B
=> [apache3 internal] load .dockerignore
=> => transferring context: 2B
=> [apache1 internal] load build definition from Dockerfile
=> => transferring dockerfile: 581B
=> [apache1 internal] load .dockerignore
=> => transferring context: 2B
=> [apache2 internal] load build definition from Dockerfile
=> => transferring dockerfile: 581B
=> [apache2 internal] load .dockerignore
=> => transferring context: 2B
=> [apache2 internal] load metadata for docker.io/library/ubuntu:22.04
=> [apache3 1/3] FROM docker.io/library/ubuntu:22.04@sha256:2b7412e6465c3c7fc5bb21d3e6f1917c167358449fe
=> [apache1 internal] load build context
=> => transferring context: 31B
=> [apache3 internal] load build context
=> => transferring context: 31B
=> [apache2 internal] load build context
=> => transferring context: 31B
=> CACHED [apache1 2/3] RUN apt-get update && apt-get install -y apache2 php php-mysql
=> CACHED [apache1 3/3] COPY index.php /var/www/html/
=> [apache1] exporting to image
=> => exporting layers
=> => writing image sha256:c95528e58d97b04ac05bcfc70bb1b9bd330e1e7c49aa84dc642a8a3ca0b26b60
=> => naming to docker.io/library/milestone1-apache1
=> [apache3] exporting to image
=> => exporting layers
=> => writing image sha256:dad8417e6429a07b54546ea9dbd80d93173886b742b3f21c498c22079502ddd3
=> => naming to docker.io/library/milestone1-apache3
=> [apache2] exporting to image
=> => exporting layers
=> => writing image sha256:ea08031161096cbabb4a729aed90da625d68b226f2c432f240841342cdf27e93
=> => naming to docker.io/library/milestone1-apache2
[+] Running 6/6
✔ Network milestone1_lampnet Created 0.3s
✔ Container contsql-m1-BD Started 0.3s
✔ Container contapa2-m1-BD-2 Started 0.2s
✔ Container contapa2-m1-BD-3 Started 0.2s
✔ Container contapa2-m1-BD-1 Started 0.1s
✔ Container milestone1-nginx-1 Started 0.1s
baisangur@user-VirtualBox:~/milestone1$

```

```
docker ps
```

gebruik ik om mijn running containers en hun info te tonen.

```

baisangur@user-VirtualBox: ~/milestone1$ docker ps
CONTAINER ID   IMAGE                                CREATED      STATUS      PORTS
2bf27b9ee41a   nginx:latest                        8 minutes ago Up 8 minutes 0.0.0.0:8085->80/tcp, :::8085->80/tcp   milestone1-nginx-1
147d00e5135f6   milestone1-apache2                 8 minutes ago Up 8 minutes 0.0.0.0:8086->80/tcp, :::8086->80/tcp   contap2-m1-B0-2
a1f64260fd9a   milestone1-apache3                 8 minutes ago Up 8 minutes 0.0.0.0:8087->80/tcp, :::8087->80/tcp   contap2-m1-B0-3
f024c6dc6c1a   milestone1-apache1                 8 minutes ago Up 8 minutes 0.0.0.0:8088->80/tcp, :::8088->80/tcp   contap2-m1-B0-1
ae40de625a5f   mysql:latest                       8 minutes ago Up 8 minutes 3306/tcp, 33060/tcp                  contsql-m1-B0
baisangur@user-VirtualBox: ~/milestone1$

```

Als ik naar de url <http://192.168.56.44:8085/index.php> in zowel mijn virtual machine als buiten mijn virtual machine dan kom ik op om te beurt via de Nginx load balancer met Round Robin op 1 van mijn 3 Apache webserver terecht. De screenshots zijn genomen buiten mijn ubuntu virtual machine. Op de screenshots is te zien dat mijn naam en de container id van elk apache container word geprint. Merk op dat de id's hetzelfde zijn als de id's in de screenshot van de docker ps output. Via mijn video zal ik bewijzen dat de naam die getoont word op de site komt uit database van de mysql container. Ook zal ik via de video tonen dat de Nginx load balancer werkt.



Baisangur - Dudayev has reached Milestone 1!!

Hello from container: f024c6d6c1a1



Baisangur - Dudayev has reached Milestone 1!!

Hello from container: 14b00e5135f6



Baisangur - Dudayev has reached Milestone 1!!

Hello from container: a1f64260fd9a

### **Link naar video**

<https://youtu.be/rOpuAOjX81Q>

## Code en commentaar

Mijn code heb ik gekopieerd naar visual studio code en vanuit visual studio code naar dit document.

### docker-compose.yaml

```
version: '3'

# Dit zijn de containers

services:

  # originele webserver container

  apache1:

    build:

      # huidige dir word genomen als build context

      context: .

      # Dockerfile zit in huidige directory, dit maakt the custom
      image aan

      dockerfile: Dockerfile

      container_name: contapa2-m1-BD-1

      # mysql service word gemaakt en gestart voor dat apache
      service start

      depends_on:

        - mysql

      # deze gegevens worden dynamisch gebruikt in index.php, zodat
      ik niet de gegevens

      # moet hardcoden in index.php

      environment:

        - MYSQL_HOST=mysql

        - MYSQL_USER=myuser
```



```

    - MYSQL_PASSWORD=mypassword

    - MYSQL_DATABASE=mydb

    # poort 8085 staat in commentaar omdat de apache webserver
    # via nginx worden bereikt met reverse_proxy

    # daarom heb ik nginx toegankelijk gemaakt op poort 8085 om
    # alle 3 webserver te bereiken met round robin

    # ports:

    # - 8085:80

    # gedeelde netwerk van containers

networks:

    - lampnet

# apache2 en apache3 zijn bedoeld voor load balancing

apache2:

    build:

        context: .

        dockerfile: Dockerfile

    container_name: contapa2-m1-BD-2

    depends_on:

        - mysql

    environment:

        - MYSQL_HOST=mysql

        - MYSQL_USER=myuser

        - MYSQL_PASSWORD=mypassword

        - MYSQL_DATABASE=mydb

```

```
# ports:

#   - 8086:80

networks:

  - lampnet

apache3:

  build:

    context: .

    dockerfile: Dockerfile

  container_name: contapa2-m1-BD-3

  depends_on:

    - mysql

  environment:

    - MYSQL_HOST=mysql

    - MYSQL_USER=myuser

    - MYSQL_PASSWORD=mypassword

    - MYSQL_DATABASE=mydb

# ports:

#   - 8087:80

networks:

  - lampnet

mysql:

  # nieuwste image van mysql gebruiken

  image: mysql:latest
```

```

    container_name: contsql-m1-BD

    # deze environment variabelen worden gebruikt bij de
    initialisatie van de mysqlcontainer

    environment:

        - MYSQL_ROOT_PASSWORD=root_password

        - MYSQL_DATABASE=mydb

        - MYSQL_USER=myuser

        - MYSQL_PASSWORD=mypassword

    volumes:

        - mysql-data2:/var/lib/mysql

        # dit is een bind volume voor de init file in mijn build
        context, het maakt de eerst table

        # met mijn naam aan

        - ./init.sql:/docker-entrypoint-initdb.d/init.sql

    # toevoegen aan gedeelde netwerk van containers

    networks:

        - lampnet

nginx:

    image: nginx:latest

    volumes:

        - ./default.conf:/etc/nginx/conf.d/default.conf

    ports:

        - 8085:80

    # toevoegen aan netwerk

```

```

    networks:

        - lampnet

    # webserver containers starten voordat de nginx container
    start

    depends_on:

        - apache1

        - apache2

        - apache3

# netwerk maken

networks:

    lampnet:

# mijn named volume definieren voor sql data persistence

volumes:

    mysql-data2:

```

## Dockerfile

```

# Ubuntu 22.04 image gebruiken

FROM ubuntu:22.04

# zorgen dat geen vragen gesteld worden tijdens installatie bij
docker compose build

ENV DEBIAN_FRONTEND=noninteractive

# Updaten van package lists, installeren van Apache2 & PHP

# -y dient om y teantwoorden als er toch y/n vragen zouden komen

```

```
RUN apt-get update && apt-get install -y apache2 php php-mysql

# pagina uit mijn build context kopiëren naar de default directory
in apache

COPY index.php /var/www/html/

# Apache2 starten in de voorgrond

CMD ["apachectl", "-D", "FOREGROUND"]
```

## index.php

```
<?php

$db_host = 'mysql'; // de container naam als host gebruiken,
aangezien ze in hetzelfde netwerk zit niet een ip in te
stellen lukt dit en hoef ik

// getenv haalt de gegevens die ik heb ingesteld in de
envirenment van de apache service

$db_user = getenv('MYSQL_USER');

$db_password = getenv('MYSQL_PASSWORD');

$db_database = getenv('MYSQL_DATABASE');

// MySQL connectie maken

$mysqli = new mysqli($db_host, $db_user, $db_password,
$db_database);

// Nakijken voor connectie errors

if ($mysqli->connect_error) {

    die("Connection failed: " . $mysqli->connect_error);
```

```
}

// query opstellen & de kolom "name" table halen

$query = "SELECT name FROM mytable";

$result = $mysqli->query($query);

// inhoud van kolom "name" printen

if ($result) {

    if ($result->num_rows > 0) {

        while ($row = $result->fetch_assoc()) {

            echo "<p>" . $row['name'] . " has reached Milestone  
1!!" . "</p>";

        }

    } else {

        echo 'No names';

    }

}

// naam van container printen

echo "Hello from container: " . gethostname();

// mysql connectie sluiten

$mysqli->close();

?>
```

## init.sql

```
-- table maken en mijn naam er in plaatsen, deze code word
uitgevoerd wanneer

-- de database word aangemaakt

CREATE TABLE mytable (

    id INT AUTO_INCREMENT PRIMARY KEY,

    name VARCHAR(255) NOT NULL

);

INSERT INTO mytable (name) VALUES ('Baisangur - Dudayev');
```

## default.conf

(C., 2017) gebruikt om http {} weg te doen.

```
# "proxy_pass http://apache;" herleid de gebruiker naar een
van deze 3 services op round robin wijze

upstream apache {

    server apache1:80;

    server apache2:80;

    server apache3:80;

}

server {

    # in de container word geluisterd op poort 80

    listen 80;

    server_name localhost;

    # als we naar deze locatie gaan bij bijvoorbeeld de url va
neen website dan worden we herleid naar "apache"
```

```
location / {  
  
    proxy_pass http://apache;  
  
}  
  
# error pagina  
  
error_page 500 502 503 504 /50x.html;  
  
location = /50x.html {  
  
    root /usr/share/nginx/html;  
  
}  
  
}
```



## Reflectie AI

Tijdens de oefeningen heb ik al veel gewerkt met Docker compose & nginx waardoor ik daar al veel kennis van had. AI heeft mij vooral geholpen met de php code & installaties in de dockerfile en om de link te leggen tussen php & de mysql database met docker compose file & de apache server. Meeste code uit de docker compose file begreep ik al door de oefeningen, maar AI heeft mij geholpen om de basis structuur van mijn project snel te maken en het heeft bepaalde code die ik niet helemaal begreep verlicht voor mij.

De antwoorden die ik gekregen heb van GPT heb ik niet alleen productief kunnen gebruiken in mijn project maar ik heb ze ook begrepen waardoor ik volgende keer minder nood zal hebben aan AI, maar doordat ik nu meer kennis heb zal ik volgende betere vragen kunnen stellen aan gpt waardoor het mij nog meer kan helpen of ik kan het gebruiken om sneller te werken.

AI kan vragen beantwoorden in een second. Als ik dezelfde vraag zou proberen op te zoeken zou het veel langer duren om op een antwoord te komen. Nog een voordeel van AI dat mij heeft geholpen is dat het specifieke vragen kan beantwoorden in een korte tijd. Dit zou je niet met een search engine kunnen doen tenzij het een algemene vraag is of als je lang genoeg zoekt. Het enigste dat dichtbij AI komt zijn forums of vragen stellen aan een leerkracht maar dit is niet te vergelijken, omdat AI ervoor heeft gezorgd dat ik zelfstandig snel kan doorwerken zonder dat ik moet wachten tot iemand mij helpt.

## Bronnen AI (chat-GPT)

### Prompt 1 help with mysql, index.php

i am working with docker compose, i have a mysql container and an apache container i want to know how to set the envirement variables in the docker compose file and the index.php file so thati can connect with my database

ChatGPT

To connect your Apache container to a MySQL container using Docker Compose, you'll need to set environment variables in your Docker Compose file for both the MySQL and Apache containers, and configure your PHP code accordingly. Here's a step-by-step guide to achieve this:

1. Create a Docker Compose file (e.g., `docker-compose.yml`) in your project directory.
2. Define your services in the Compose file. You'll need two services: one for the MySQL container and one for the Apache container. Below is a minimal example of a `docker-compose.yml` file:

```
yaml
version: '3'
services:
  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: your_root_password
      MYSQL_DATABASE: your_database_name
      MYSQL_USER: your_username
      MYSQL_PASSWORD: your_password

  apache:
    image: httpd:2.4
    ports:
      - "80:80"
    depends_on:
      - mysql
    environment:
      MYSQL_HOST: mysql
      MYSQL_DATABASE: your_database_name
      MYSQL_USER: your_username
      MYSQL_PASSWORD: your_password
```

In the above Compose file, we define two services: `mysql` and `apache`. The `mysql` service uses the official MySQL 5.7 image and sets environment variables for the MySQL server. The `apache` service uses the official Apache 2.4 image and depends on the `mysql` service. It also sets environment variables to be used by your PHP code to connect to the MySQL server.

3. In your PHP code (e.g., `index.php`), you can use these environment variables to connect to the MySQL database. Here's an example of how to do it using PHP and the `mysqli` extension:

```
php
<?php
$host = getenv('MYSQL_HOST');
```

```

$database = getenv('MYSQL_DATABASE');
$username = getenv('MYSQL_USER');
$password = getenv('MYSQL_PASSWORD');

$mysqli = new mysqli($host, $username, $password, $database);

if ($mysqli->connect_error) {
    die("Connection failed: " . $mysqli->connect_error);
}

echo "Connected to the MySQL database!";
$mysqli->close();
?>

```

Make sure you have the `mysqli` extension enabled in your Apache container. You can create a custom Docker image for your Apache container with the necessary extensions or install them at runtime using a Dockerfile.

4. Run your Docker Compose setup with the following command from your project directory:

```

bash
docker-compose up

```

This command will start both the MySQL and Apache containers with the specified environment variables. Your PHP code in the Apache container can now connect to the MySQL container using these variables.

## Prompt 2 print table values

i have a table namd "mytable" with the column "name". How do i print every name in the table using the code in index.php above

ChatGPT

To print every name from the "mytable" table with the "name" column in your MySQL database using the PHP code provided in your `index.php`, you can use SQL queries to fetch and display the data. Here's how you can modify your `index.php` to accomplish this:

```

php
<?php
$host = getenv('MYSQL_HOST');
$database = getenv('MYSQL_DATABASE');
$username = getenv('MYSQL_USER');
$password = getenv('MYSQL_PASSWORD');

$mysqli = new mysqli($host, $username, $password, $database);

if ($mysqli->connect_error) {
    die("Connection failed: " . $mysqli->connect_error);
}

$query = "SELECT name FROM mytable";
$result = $mysqli->query($query);

if ($result->num_rows > 0) {

```

```

        echo "Names in the 'mytable' table:<br>";
        while ($row = $result->fetch_assoc()) {
            echo $row['name'] . "<br>";
        }
    } else {
        echo "No names found in the 'mytable' table.";
    }

    $mysqli->close();
?>

```

In the code above, we first establish a connection to the MySQL database using the provided environment variables. Then, we execute an SQL query to retrieve all the names from the "mytable" table. We use a `SELECT` statement to fetch the "name" column.

If there are rows returned from the query, it means there are names in the table, and we loop through the result set and print each name. If there are no names, it will display a message indicating that.

Make sure the `mytable` and `name` column exist in your MySQL database and are accessible by the MySQL user you specified in the environment variables.

### Prompt 3 create init.sql

how to set up tables to mysql docker

ChatGPT

To set up tables in a MySQL Docker container, you can use one of the following methods:

#### Method 1: Use a SQL Initialization Script

1. Create a `.sql` file that contains the SQL commands to create your tables. For example, you can create a file named `init.sql` with the following content:

```

sql
-- Create a table named 'mytable'
CREATE TABLE mytable (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255)
);

-- Add some sample data
INSERT INTO mytable (name) VALUES
    ('John'),
    ('Alice'),
    ('Bob');

```

2. Modify your Docker Compose file (`docker-compose.yml`) to include a volume that mounts the SQL initialization script into the MySQL container. Add a `volumes` section under the `mysql` service definition:

```

yaml
version: '3'

```

```

services:
  mysql:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: your_root_password
      MYSQL_DATABASE: your_database_name
      MYSQL_USER: your_username
      MYSQL_PASSWORD: your_password
    volumes:
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql

```

In the `volumes` section, we map the local `init.sql` file to `/docker-entrypoint-initdb.d/init.sql` inside the MySQL container. MySQL automatically runs the scripts in this directory when the container starts.

3. Save the changes to your `docker-compose.yml` and then run the following command to start your containers:

```

bash
docker-compose up

```

This will start the MySQL container and execute the SQL initialization script, creating the table and inserting sample data.

## Method 2: Use MySQL Client Inside the Container

If you prefer to set up tables interactively, you can enter the MySQL container using the MySQL client and execute SQL commands manually. Follow these steps:

1. Start your MySQL container:

```

bash
docker-compose up -d

```

2. Enter the MySQL container using the `mysql` command. You need to specify the MySQL username, password, and the database name. In your Docker Compose configuration, you defined these in the `MYSQL_USER`, `MYSQL_PASSWORD`, and `MYSQL_DATABASE` environment variables.

```

bash
docker exec -it your_mysql_container_name mysql -u your_username
-p your_database_name

```

Replace `your_mysql_container_name`, `your_username`, and `your_database_name` with the appropriate values.

3. You'll be prompted to enter the MySQL password. Once inside the MySQL client, you can create tables and insert data using SQL commands. For example:

```

sql

```

```
CREATE TABLE mytable (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255)
);

INSERT INTO mytable (name) VALUES
  ('John'),
  ('Alice'),
  ('Bob');
```

After setting up your tables and data, you can exit the MySQL client by typing `exit`.

By using one of these methods, you can set up tables in your MySQL Docker container. The first method is more automated and suitable for scripted deployments, while the second method allows for manual interaction within the container.

## Prompt 4 non interactive installation

during the installation of docker build i get prompt but i can't answer them.  
How can i skip these questions in my dockerfile

ChatGPT

If you're running into prompts during the installation of a package or software when building a Docker image and you want to automate the installation process without interaction, you can use environment variables or command-line options to set non-interactive mode.

It's important to remember that the exact way to achieve this might vary depending on the specific package or software you are installing. Here are some common strategies:

1. **Environment Variables:** Many installation tools provide environment variables that can be used to set non-interactive mode. For example, for `apt-get` (used in Debian/Ubuntu-based images), you can use the `DEBIAN_FRONTEND` variable to set non-interactive mode.

```
dockerfile
• ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get -y install your-package
```

Be sure to check the documentation of the specific tool you're using to see if it provides such variables.

- **Command-Line Options:** Some installation commands have command-line options that allow you to specify non-interactive mode. For example, with `apt-get`, you can use the `-y` or `--yes` option to assume "yes" for all prompts.

```
dockerfile
• RUN apt-get -y install your-package
```

- **Pre-configure Packages:** If you're installing a package that has a configuration script, you can pre-configure it by providing the necessary configuration settings as part of your Dockerfile.

```
dockerfile
• RUN echo 'Package configuration setting' | debconf-set-
  selections
RUN apt-get -y install your-package
```

- **Silence Output:** Sometimes, you may need to redirect or suppress output to prevent prompts from being displayed. For example, you can use `>/dev/null` to redirect output to `/dev/null`.

```
dockerfile
  4. RUN apt-get -y install your-package >/dev/null
  5.
```

Remember to consult the documentation for the specific package or software you're installing to find out if there are any additional steps required to automate the installation process. The approach may vary depending on the software and the package manager being used in your Docker image.

## Prompt 5 set containername

how do i change the containername in my docker compose file?

ChatGPT

To change the name of a container defined in your Docker Compose file, you can use the `container_name` property. Here's how you can do it:

Suppose you have a Docker Compose file (e.g., `docker-compose.yml`) with a service named "my-service" and you want to change the name of the container for that service.

Here's an example of a Docker Compose file:

```
yaml
version: '3'
services:
  my-service:
    image: your-image:tag
    ports:
      - "8080:80"
```

To set a custom name for the container, you can add the `container_name` property under the service definition, like this:

```
yaml
version: '3'
services:
  my-service:
    image: your-image:tag
    ports:
      - "8080:80"
    container_name: my-custom-container
```

```
- "8080:80"
container_name: my-custom-container-name
```

In this example, I added `container_name: my-custom-container-name` under the `my-service` definition. Replace `my-custom-container-name` with the desired name for your container.

Once you've made this change, when you start your containers using `docker-compose up`, the container for "my-service" will have the custom name you specified.

Remember that container names must be unique on your Docker host. If you're running multiple instances of the same Docker Compose file or containers with the same names already exist, you may need to stop or remove the existing containers to avoid conflicts.

## Prompt 6, 7, 8 load balancing with nginx

i making horizontal load balancing for my apache webserver with docker compose using nginx & reverse\_proxy. I am trying to make use of round robin for this.

ChatGPT

Horizontal load balancing with Docker Compose, Nginx, and a reverse proxy like NGINX can help distribute incoming web traffic across multiple Apache web server containers in a round-robin fashion. Here's a step-by-step guide on how to set this up:

1. **Docker Compose Setup:** Start by creating a `docker-compose.yml` file that defines your services. In this example, we'll set up two Apache web server containers:

```
yaml
• version: '3'
services:
  apache1:
    image: httpd
    container_name: apache1
    ports:
      - "8080:80"
  apache2:
    image: httpd
    container_name: apache2
    ports:
      - "8081:80"
```

This configuration creates two Apache containers, each listening on different ports.

- **NGINX Reverse Proxy:** Add an NGINX reverse proxy service to distribute requests to the Apache containers. This service will use the `nginx:latest` image.

yaml



```

nginx:
  image: nginx:latest
  container_name: nginx
  ports:
    - "80:80"
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf
  depends_on:
    - apache1
    - apache2

```

You also need to create an `nginx.conf` file that configures the NGINX reverse proxy for round-robin load balancing. Here's a sample configuration:

```

nginx
• events {}

http {
    upstream apache_servers {
        server apache1:80;
        server apache2:80;
        # Add more servers if needed
    }

    server {
        listen 80;
        location / {
            proxy_pass http://apache_servers;
        }
    }
}

```

This configuration defines a round-robin load balancing strategy for distributing requests to the Apache containers.

- **Run Docker Compose:** Navigate to the directory containing your `docker-compose.yml` file and start your services:

```

bash
3. docker-compose up -d
4. This will start the Apache and NGINX containers in the background.
5. Testing: You can now access your web application through the NGINX reverse proxy. Open a web browser and navigate to http://localhost (or the IP address of your server). NGINX will distribute incoming requests in a round-robin manner to your Apache containers.

```

That's it! You've set up a basic horizontal load balancing configuration using Docker Compose, NGINX, and Apache web server containers. You can scale the number of Apache containers as needed and update the NGINX configuration accordingly for more advanced load balancing scenarios.

why events { }

ChatGPT

In NGINX configuration, the `events` block is used to specify various settings related to how NGINX handles network connections and events. However, in the provided example for a simple NGINX reverse proxy with round-robin load balancing, the `events` block is left empty with just `events {}`.

In a basic load balancing setup, the `events` block is often empty or contains minimal configuration because it's not typically used for configuring event handling parameters. NGINX can handle a wide range of event-related configurations, but in this specific scenario, you don't need to specify any event-specific settings, so an empty `events` block is sufficient.

If you were setting up more complex configurations, such as fine-tuning NGINX's event handling behavior for high-performance scenarios or dealing with specific network considerations, you might need to include options within the `events` block. For many typical use cases, though, an empty `events` block is perfectly acceptable.

should i leave the ports for the apache services open?

ChatGPT

In a Docker Compose setup where you have a reverse proxy (such as NGINX) in front of your Apache web server containers, you don't necessarily need to expose the Apache container ports to the host. Exposing ports directly to the host is more common when you want to access a service directly from the host machine, but in this scenario, you want to route traffic through the reverse proxy.

Here's how you can manage the ports for the Apache services:

1. **Exposing Ports:** If you want to access the Apache web server containers directly from the host machine for debugging or other purposes, you can expose the ports as you did in your original `docker-compose.yml` file. This allows you to access each Apache container individually via `http://localhost:8080` and `http://localhost:8081`, respectively. This is useful if you want to perform tasks like testing or debugging Apache directly.

```
    yaml
  • services:
    apache1:
      image: httpd
      container_name: apache1
      ports:
        - "8080:80"
    apache2:
      image: httpd
      container_name: apache2
      ports:
        - "8081:80"
```

- **Not Exposing Ports:** If your primary goal is to route incoming web traffic through the NGINX reverse proxy and you don't need to access the Apache

containers directly from the host, you can remove the port mappings. This means the Apache containers will only be accessible from within the Docker network. In this case, your `docker-compose.yml` file for the Apache services might look like this:

```
yaml
2. services:
3.   apache1:
4.     image: httpd
5.     container_name: apache1
6.   apache2:
7.     image: httpd
8.     container_name: apache2
9.
```

The decision to expose or not expose ports for the Apache services depends on your specific use case. If you don't have a need to access them directly and the sole purpose is to route traffic through NGINX for load balancing, not exposing the ports can enhance the security of your setup by limiting direct access to the Apache containers.

## Bronnen

C., S. (2017, april 26). *nginx: [emerg] "http" directive is not allowed in /etc/nginx/sites-enabled/default:1* - *Stack Overflow*. Opgeroepen op oktober 30, 2023, van [stackoverflow.com](https://stackoverflow.com/questions/43643829/nginx-emerg-http-directive-is-not-allowed-here-in-etc-nginx-sites-enabled):  
<https://stackoverflow.com/questions/43643829/nginx-emerg-http-directive-is-not-allowed-here-in-etc-nginx-sites-enabled>