

Milestone 1 - Docker Compose Stack Documentation

Table of Contents

1. [Overview](#)
 2. [Architecture](#)
 3. [Prerequisites](#)
 4. [Project Structure](#)
 5. [Configuration Files](#)
 6. [Environment Variables](#)
 7. [Step-by-Step Deployment Guide](#)
 8. [Service Descriptions](#)
 9. [Verification and Testing](#)
-

Overview

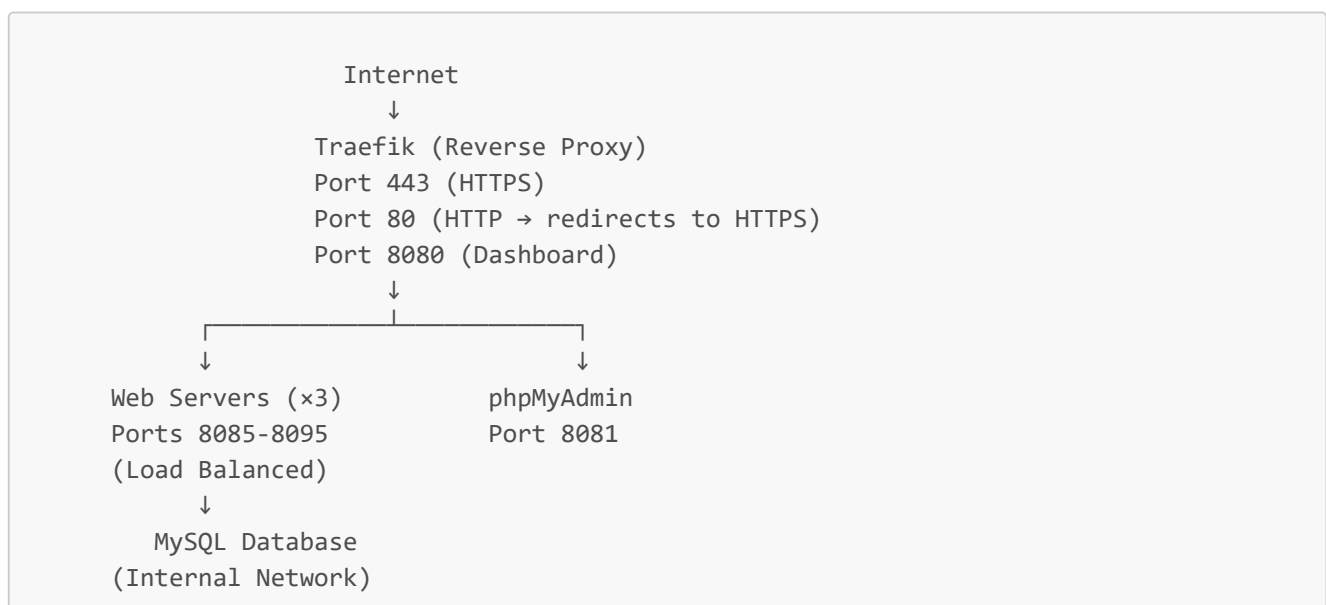
This Docker Compose stack creates a scalable web application infrastructure with the following components:

- **Traefik:** Reverse proxy with automatic HTTPS/SSL certificate management
- **Web Servers (Apache/PHP):** Three replicas of Apache web servers with PHP-FPM support
- **MySQL Database:** Persistent database backend
- **phpMyAdmin:** Web-based database administration tool

The stack demonstrates:

- Load balancing across multiple web server instances
 - Automatic SSL certificate provisioning via Let's Encrypt
 - Container health monitoring
 - Service dependencies and startup ordering
 - Persistent data storage
-

Architecture



Prerequisites

Before deploying this stack, ensure you have:

1. Docker Engine

- Check version: `docker --version`

2. Docker Compose

- Check version: `docker compose version`

3. Operating System:

- Linux (recommended for production)
- Windows with WSL2 or Docker Desktop
- macOS with Docker Desktop

4. Domain Name (for production with real SSL certificates)

- DNS properly configured to point to your server's IP

5. Network Requirements:

- Ports 80, 443, 8080, 8085-8095, 8081 available
 - Internet connection for pulling images and obtaining SSL certificates
-

Project Structure

```
Milestone 1/
├── docker-compose.yml      # Main orchestration file
├── .env.example            # Copy to .env and fill in for production
├── webserver/              # Web server configuration
│   ├── Dockerfile          # Custom Apache/PHP image
│   ├── 000-default.conf    # Apache VirtualHost configuration
│   ├── init.sql            # Database initialization script
│   └── www/                 # Web application files
│       └── index.php        # Main application file
```

Configuration Files

docker-compose.yml

The main orchestration file that defines all services, networks, and volumes.

Key sections explained:

- **services**: Defines each container/service
- **volumes**: Named volumes for persistent data

- `depends_on`: Service startup dependencies
- `healthcheck`: Container health monitoring
- `labels`: Traefik routing configuration

Dockerfile

Custom image based on Ubuntu 24.04 with:

- Apache web server
- PHP 8.3 with FPM (FastCGI Process Manager)
- MySQL PHP extension
- Security hardening (non-root user)

000-default.conf

Apache VirtualHost configuration that:

- Sets up document root at `/var/www/html`
- Configures PHP-FPM integration via Unix socket
- Enables directory permissions

init.sql

Database initialization script that:

- Creates a `users` table
- Inserts sample data

index.php

PHP application that:

- Connects to MySQL database
- Displays user data
- Shows which container is serving the request (for load balancing verification)

Environment Variables

Copy `.env.example` to `.env` in the `Milestone 1/` directory with the following variables:

```
# Email for Let's Encrypt notifications
EMAIL=your-email@example.com

# Domain name for your application
DOMAIN=yourdomain.com

# ACME Server (use staging for testing)
# Staging: https://acme-staging-v02.api.letsencrypt.org/directory
# Production: https://acme-v02.api.letsencrypt.org/directory
ACME_SERVER=https://acme-staging-v02.api.letsencrypt.org/directory

# MySQL Configuration
```

```
MYSQL_ROOT_PASSWORD=secure_root_password_here
MYSQL_DATABASE=milestone_db
MYSQL_USER=milestone_user
MYSQL_PASSWORD=secure_user_password_here
```

Parameter Explanations:

Variable	Description	Example
EMAIL	Email address for Let's Encrypt certificate notifications	admin@example.com
DOMAIN	Your domain name for the web application	example.com or www.example.com
ACME_SERVER	Let's Encrypt API endpoint (staging or production)	Use staging for testing
MYSQL_ROOT_PASSWORD	MySQL root user password (keep secure!)	Strong password with special chars
MYSQL_DATABASE	Name of the database to create	milestone_db
MYSQL_USER	MySQL user for the application	milestone_user
MYSQL_PASSWORD	Password for the MySQL user	Strong password

Step-by-Step Deployment Guide

Step 1: Clone or Navigate to Project Directory

```
cd ~/"Milestone 1"
```

Explanation: Navigate to the directory containing the `docker-compose.yml` file.

Step 2: Create Environment File

Create a `.env` file with your configuration:

```
cp .env.example .env && vim .env
```

Explanation: This opens vim to create the environment file. Fill in the environment variables from the [Environment Variables](#) section and save.

Screenshot:

```
# Database Configuration
MYSQL_ROOT_PASSWORD=rooti
MYSQL_DATABASE=appdbi
MYSQL_USER=appuseri
MYSQL_PASSWORD=secreti

# ACME/Let's Encrypt Configuration
DOMAIN=testing.davidmaat.be
EMAIL=maatdavid@yahoo.com
ACME_SERVER=https://acme-v02.api.letsencrypt.org/directory
```

Step 3: Validate Docker Compose Configuration

```
docker compose config
```

Explanation: This command validates the syntax of your `docker-compose.yml` file and shows the merged configuration with environment variables substituted.

What to look for:

- No error messages
- Environment variables properly substituted
- All services listed correctly

Partial screenshot:

```
replicas: 3
environment:
  COMPOSE_PROJECT_NAME: milestone1
  MYSQL_DATABASE: appdbi
  MYSQL_HOST: contsql-ml-dm
  MYSQL_PASSWORD: secreti
  MYSQL_USER: appuseri
healthcheck:
  test:
    - CMD
    - curl
    - -f
    - -s
    - http://localhost/
  timeout: 10s
  interval: 30s
  retries: 3
  start_period: 10s
labels:
  traefik.enable: "true"
  traefik.http.routers.web.entrypoints: websecure
  traefik.http.routers.web.rule: Host(`testing.davidmaat.be`)
  traefik.http.routers.web.tls.certresolver: letsencrypt
  traefik.http.services.web.loadbalancer.server.port: "80"
networks:
  default: null
ports:
  - mode: ingress
    target: 80
    published: 8085-8095
    protocol: tcp
volumes:
  - type: bind
    source: /home/david/lwsml/linux-web-services/Milestone 1/webserver/www
    target: /var/www/html
    bind:
      create_host_path: true
mysql:
  container_name: contsql-ml-dm
  environment:
    MYSQL_DATABASE: appdbi
    MYSQL_PASSWORD: secreti
    MYSQL_ROOT_PASSWORD: rooti
    MYSQL_USER: appuseri
```

Step 4: Pull Required Docker Images

```
docker compose pull
```

Explanation: Downloads all required Docker images before building. This separates the download process from the build process.

Images pulled:

- `traefik:v3.2` (Reverse proxy)
- `mysql:8.0` (Database)
- `phpmyadmin:latest` (Database admin tool, latest is okay here, since it's not exposed publicly)
- `ubuntu:24.04` (Base image for custom web server)

```
david@SRVDAVID:~/lwsml/linux-web-services/Milestone 1$ sudo docker compose pull
[+] Pulling 4/4
✔ contapa2-ml-dm Skipped - No image to be pulled
✔ phpmyadmin Pulled
✔ traefik Pulled
✔ mysql Pulled
```

```
docker compose build
```

- Installing Apache and PHP
- Configuring PHP-FPM
- Setting up a non-root user for security
- Copying application files

Instruction	Purpose
FROM ubuntu:24.04@sha256:...	Base image with specific digest for reproducibility
RUN apt update && apt install...	Install Apache, PHP, and required extensions
RUN useradd -r -u 1001...	Create non-root user for running Apache (security best practice)
COPY ./www /var/www/html	Copy application files into the image
RUN chown -R apacheuser:www-data...	Set proper file permissions
EXPOSE 80	Document that container listens on port 80
CMD ["/start.sh"]	Default command to run when container starts

```
[+]  
[+] Building 1.2s (13/13) FINISHED  
-> [contpaas-m] internal load build definition from Dockerfile 0.02  
-> << transferring context file 4:338 0.04  
[contpaas-m] --rm internal load metadata for docker.io/library/ubuntu:24.04base361933c76eaa1bbad3c62b3d7d7ec780ca05bcca1e4f7ea5dbda33a2d166fc 0.88  
-> [contpaas-m] internal load .dockerignore 0.02  
-> << transferring context file 2B 0.04  
[contpaas-m] --rm 1/7 FROM docker.io/library/ubuntu:24.04base361933c76eaa1bbad3c62b3d7d7ec780ca05bcca1e4f7ea5dbda33a2d166fc 0.08  
-> [contpaas-m] internal load build context 0.04  
-> << transferring context file 848 0.04  
CACHED [contpaas-m] 2/7 RUN apt update && apt install -y apache2 php-fpm php-mysql curl && a2enmod proxy_fcgi proxy && rm -rf /var/www/html/* 0.08  
CACHED [contpaas-m] 3/7 RUN useradd -s /bin/bash www-data && www-data && echo www-data && passwd www-data && sed -i 's/#export APACHE_RUN_USER=www-data/export APACHE_RUN_USER=apacheuser/' /etc/apache2/envvars && sed -i 's/#export APACHE_RUN_GROUP=www-data/export APACHE_RUN_GROUP=apache' /etc/apache2/envvars && systemctl enable httpd && systemctl start httpd && systemctl restart httpd && systemctl restart php-fpm && systemctl restart mysql && 0.04  
CACHED [contpaas-m] 4/7 COPY ./src /var/www/html 0.04  
CACHED [contpaas-m] 5/7 RUN chown -R apache:apache www-data /var/www/html && sed -i 's/export APACHE_RUN_USER=www-data/export APACHE_RUN_USER=apacheuser/' /etc/apache2/envvars && sed -i 's/export APACHE_RUN_GROUP=www-data/export APACHE_RUN_GROUP=apache' /etc/apache2/envvars && systemctl enable httpd && systemctl start httpd && systemctl restart httpd && systemctl restart php-fpm && systemctl restart mysql && 0.04  
CACHED [contpaas-m] 6/7 RUN apt-get update && apt-get install -y libapache2-mod-php && echo '#!/bin/sh\nsha256sum=$(sha256sum $(cat /dev/urandom | fold -n 1024 | shuf -v | paste -sd ''\n'' -))\necho \"$0\" \"$sha256sum\"' >> /etc/php/8.1/fpm/pool.d/www.conf\nsha256sum=$(sha256sum $(cat /dev/urandom | fold -n 1024 | shuf -v | paste -sd ''\n'' -))\necho \"$0\" \"$sha256sum\"' >> /etc/php/8.1/fpm/pool.d/www.conf && 0.04  
CACHED [contpaas-m] 7/7 RUN apt-get update && apt-get install -y libapache2-mod-php && echo '#!/bin/sh\nsha256sum=$(sha256sum $(cat /dev/urandom | fold -n 1024 | shuf -v | paste -sd ''\n'' -))\necho \"$0\" \"$sha256sum\"' >> /etc/php/8.1/fpm/pool.d/www.conf\nsha256sum=$(sha256sum $(cat /dev/urandom | fold -n 1024 | shuf -v | paste -sd ''\n'' -))\necho \"$0\" \"$sha256sum\"' >> /etc/php/8.1/fpm/pool.d/www.conf && 0.04  
-> << exporting layers 0.04  
-> writing image sha256:f4de163cf799efeb0621e3a72cd0dd13ca7b3dea13ff191350cda877b0f88 0.08  
-> naming to docker.io/library/milestone1-contpaas-m 0.04  
-> [contpaas-m] pushing image milestone1-contpaas-m 0.04
```

```
docker compose up -d
```

Explanation: Starts all services in detached mode (runs in background).

Flags explained:

- **-d** or **--detach**: Run containers in the background

What happens:

1. Creates a custom Docker network for the services
2. Creates named volumes for persistent data
3. Starts MySQL container first (dependency)
4. Waits for MySQL health check to pass
5. Starts web server containers (3 replicas)
6. Starts phpMyAdmin
7. Starts Traefik reverse proxy

Screenshot:

```
david@SRVDAVID:~/lwsml/linux-web-services/Milestone 1$ sudo docker compose up -d
[+] Running 6/6
✔ Container contsql-m1-dm          Healthy
✔ Container phpmyadmin-m1-dm       Running
✔ Container milestone1-contapa2-m1-dm-1 Healthy
✔ Container milestone1-contapa2-m1-dm-2 Healthy
✔ Container milestone1-contapa2-m1-dm-3 Healthy
✔ Container traefik-m1-dm          Running
```

Step 7: Monitor Container Status

```
docker compose ps
```

Explanation: Shows the status of all containers in the stack.

Status indicators:

- **Up (healthy)**: Container is running and passed health checks
- **Up (health: starting)**: Container is running but health check hasn't passed yet
- **Exit**: Container has stopped

Screenshot:

```
david@SRVDAVID:~/lwsml/linux-web-services/Milestone 1$ sudo docker compose ps
NAME                IMAGE                                COMMAND                                SERVICE    CREATED     STATUS      PORTS
contsql-m1-dm       mysqldb@sha256:5367102acfeaa47eb0eb57c8df8b96c8c14004559131eac9bbfaa62f01e34 "/start.sh"  contsql-m1-dm  13 hours ago Up 13 hours (healthy) 3306/tcp, 33060/tcp
milestone1-contapa2-m1-dm-1 milestone1-contapa2-m1-dm "/start.sh"  contapa2-m1-dm  13 hours ago Up 13 hours (healthy) 0.0.0.0:8087->80/tcp, [::]:8087->80/tcp
milestone1-contapa2-m1-dm-2 milestone1-contapa2-m1-dm "/start.sh"  contapa2-m1-dm  13 hours ago Up 13 hours (healthy) 0.0.0.0:8087->80/tcp, [::]:8087->80/tcp
milestone1-contapa2-m1-dm-3 milestone1-contapa2-m1-dm "/start.sh"  contapa2-m1-dm  13 hours ago Up 13 hours (healthy) 0.0.0.0:8087->80/tcp, [::]:8087->80/tcp
phpmyadmin-m1-dm    phpmyadmin/phpmyadmin              "/docker-entrypoint..." phpmyadmin     13 hours ago Up 13 hours      0.0.0.0:8081->80/tcp, [::]:8081->80/tcp
traefik-m1-dm       traefik:v3.2                        "/entrypoint.sh --ap..." traefik        13 hours ago Up 13 hours (healthy) 0.0.0.0:80->80/tcp, ::80->80/tcp, 0.0.0.0:443->443/tcp, ::443->443/tcp, 0.0.0.0:8080->8080/tcp, ::8080->8080/tcp
```

Step 8: View Container Logs

Check logs for all services:

```
docker compose logs
```

Check logs for a specific service:

```
docker compose logs traefik
```

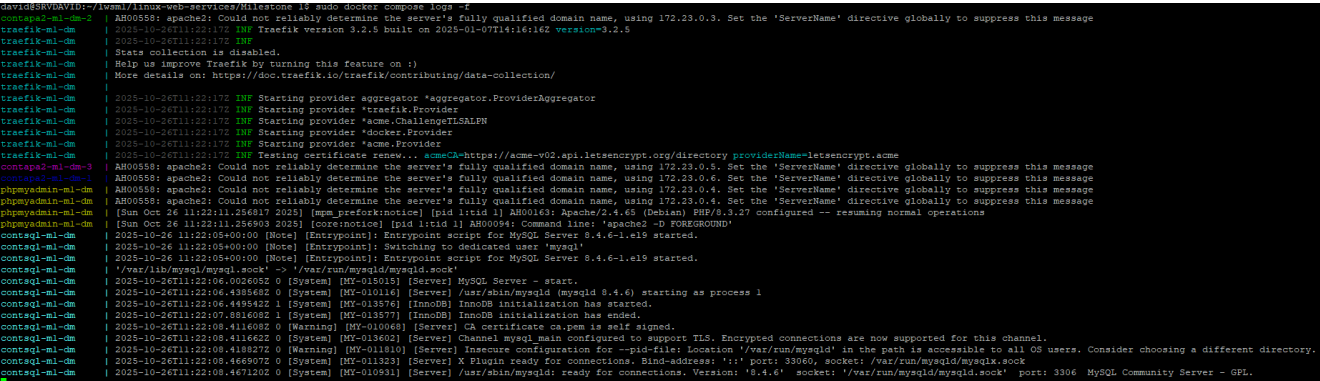
Follow logs in real-time:

```
docker compose logs -f contapa2-m1-dm
```

Flags explained:

- f or --follow: Stream logs in real-time (like tail -f)
- tail=100: Show only last 100 lines
- timestamps: Add timestamps to log entries

Screenshot:



Service Descriptions

1. Traefik (Reverse Proxy)

Image: traefik:v3.2

Purpose: Acts as a reverse proxy and load balancer with automatic HTTPS certificate management.

Ports Exposed:

- 80: HTTP entry point (redirects to HTTPS)
- 443: HTTPS entry point
- 8080: Traefik dashboard (insecure mode for development)

Command-line Parameters Explained:

Parameter	Explanation
--api.dashboard=true	Enable the Traefik web dashboard
--api.insecure=true	Allow dashboard access without authentication (dev only)

Parameter	Explanation
<code>--ping=true</code>	Enable ping endpoint for health checks
<code>--entrypoints.web.address=:80</code>	Create HTTP entry point on port 80
<code>--entrypoints.web.http.redirects.entrypoint.to=websecure</code>	Redirect HTTP to HTTPS
<code>--entrypoints.web.http.redirects.entrypoint.scheme=https</code>	Use HTTPS scheme for redirects
<code>--entrypoints.web.http.redirects.entrypoint.permanent=true</code>	Use HTTP 301 (permanent redirect)
<code>--entrypoints.websecure.address=:443</code>	Create HTTPS entry point on port 443
<code>--certificatesresolvers.letsencrypt.acme.email=\${EMAIL}</code>	Email for Let's Encrypt notifications
<code>--certificatesresolvers.letsencrypt.acme.storage=/letsencrypt/acme.json</code>	Where to store certificates
<code>--certificatesresolvers.letsencrypt.acme.caserver=\${ACME_SERVER}</code>	Let's Encrypt server (staging/prod)
<code>--certificatesresolvers.letsencrypt.acme.httpchallenge.entrypoint=web</code>	Use HTTP-01 challenge on port 80
<code>--providers.docker=true</code>	Enable Docker provider
<code>--providers.docker.endpoint=unix:///var/run/docker.sock</code>	Docker socket location
<code>--providers.docker.exposedbydefault=false</code>	Require explicit <code>traefik.enable=true</code> label
<code>--log.level=INFO</code>	Set logging verbosity

Volumes:

- `/var/run/docker.sock:/var/run/docker.sock:ro`: Docker socket (read-only) for service discovery
- `traefik-certs:/letsencrypt`: Persistent storage for SSL certificates

Health Check:

```
test: ["CMD", "traefik", "healthcheck", "--ping"]
interval: 5s      # Check every 5 seconds
timeout: 3s       # Fail if check takes > 3 seconds
retries: 3        # Retry 3 times before marking unhealthy
start_period: 5s  # Grace period before starting checks
```

Dependencies:

- Depends on `contapa2-m1-dm` being healthy before starting

2. Web Server (contapa2-m1-dm)

Build Context: `./webserver/Dockerfile`

Purpose: Apache web server with PHP-FPM serving the application.

Ports Exposed:

- `8085-8095:80`: Maps host ports 8085-8095 to container port 80 for 3 replicas

Deployment:

```
deploy:
  replicas: 3 # Run 3 instances (by default) of this service
```

Explanation: Docker Compose will start 3 containers from this service definition. Each gets a unique name (contapa2-m1-dm-1, contapa2-m1-dm-2, contapa2-m1-dm-3).

Environment Variables:

Variable	Purpose	Value Source
<code>MYSQL_HOST</code>	Database hostname	Hardcoded: <code>contsql-m1-dm</code>
<code>MYSQL_DATABASE</code>	Database name	From <code>.env</code> file
<code>MYSQL_USER</code>	Database username	From <code>.env</code> file
<code>MYSQL_PASSWORD</code>	Database password	From <code>.env</code> file

Volumes:

- `./webserver/www:/var/www/html`: Bind mount for live code updates (development)

Traefik Labels:

Label	Explanation
<code>traefik.enable=true</code>	Enable Traefik routing for this service
<code>traefik.http.routers.web.rule=Host(`\${DOMAIN}`)</code>	Route requests matching this domain
<code>traefik.http.routers.web.entrypoints=websecure</code>	Use HTTPS entry point
<code>traefik.http.routers.web.tls.certresolver=letsencrypt</code>	Use Let's Encrypt for SSL
<code>traefik.http.services.web.loadbalancer.server.port=80</code>	Backend container port

Health Check:

```
test: ["CMD", "curl", "-f", "-s", "http://localhost/"]
interval: 30s      # Check every 30 seconds
timeout: 10s       # Fail if check takes > 10 seconds
retries: 3         # Retry 3 times before marking unhealthy
start_period: 10s  # Wait 10s before starting checks
```

Explanation: Uses `curl` to test if the web server responds on port 80.

Dependencies:

- Depends on `mysql` being healthy before starting

3. MySQL Database

Image: `mysql:1ts@sha256:5367102acfeaea47eb0eb57c8d4f8b96c8c14004859131eac9bbfaa62f81e34`

Purpose: Persistent relational database backend.

Image Digest Explanation:

- Using `@sha256:...` pins the exact image version for reproducibility
- `1ts` tag points to Long Term Support version of MySQL

Container Name: `contsql-m1-dm`

Restart Policy: `always` - Container restarts automatically if it stops

Environment Variables:

Variable	Purpose
<code>MYSQL_ROOT_PASSWORD</code>	Password for MySQL root user
<code>MYSQL_DATABASE</code>	Database to create on initialization
<code>MYSQL_USER</code>	Non-root user to create
<code>MYSQL_PASSWORD</code>	Password for the non-root user

Health Check:

```
test: ["CMD-SHELL", "mysqladmin ping -h localhost -u${MYSQL_USER} -p${MYSQL_ROOT_PASSWORD} || exit 1"]
interval: 10s      # Check every 10 seconds
timeout: 5s        # Fail if check takes > 5 seconds
retries: 5         # Retry 5 times before marking unhealthy
start_period: 30s  # Wait 30s before starting (MySQL needs time to initialize)
```

Explanation: Uses `mysqladmin ping` to verify MySQL is accepting connections.

Volumes:

Volume	Purpose
<code>db_data:/var/lib/mysql</code>	Persistent storage for database files
<code>./webserver/init.sql:/docker-entrypoint-initdb.d/init.sql</code>	Initialization script

Initialization Process:

1. MySQL container starts
2. Creates database specified in `MYSQL_DATABASE`
3. Creates user specified in `MYSQL_USER`
4. Runs all `.sql` files in `/docker-entrypoint-initdb.d/`
5. Sets root password

4. phpMyAdmin

Image: `phpmyadmin:latest`

Purpose: Web-based database administration interface.

Container Name: `phpmyadmin-m1-dm`

Ports Exposed:

- `8081:80`: Access phpMyAdmin at `http://localhost:8081`

Environment Variables:

Variable	Value	Purpose
<code>PMA_HOST</code>	<code>mysql</code>	Hostname of MySQL server to connect to

Explanation: `PMA_HOST` uses Docker's internal DNS to resolve the MySQL service name to its IP address.

Restart Policy: `always` - Ensures phpMyAdmin is always available

Verification and Testing

1. Access Traefik Dashboard

Open your browser and navigate to:

```
http://SERVER_IP:8080
```

What to check:

- HTTP routers are configured
- All backend servers are listed
- Health status is green

David Maat has reached Milestone 1!!

Served by container: f91f92f8a953

Screenshot:

2. Access Web Application (Local)

Test direct access to individual web server replicas:

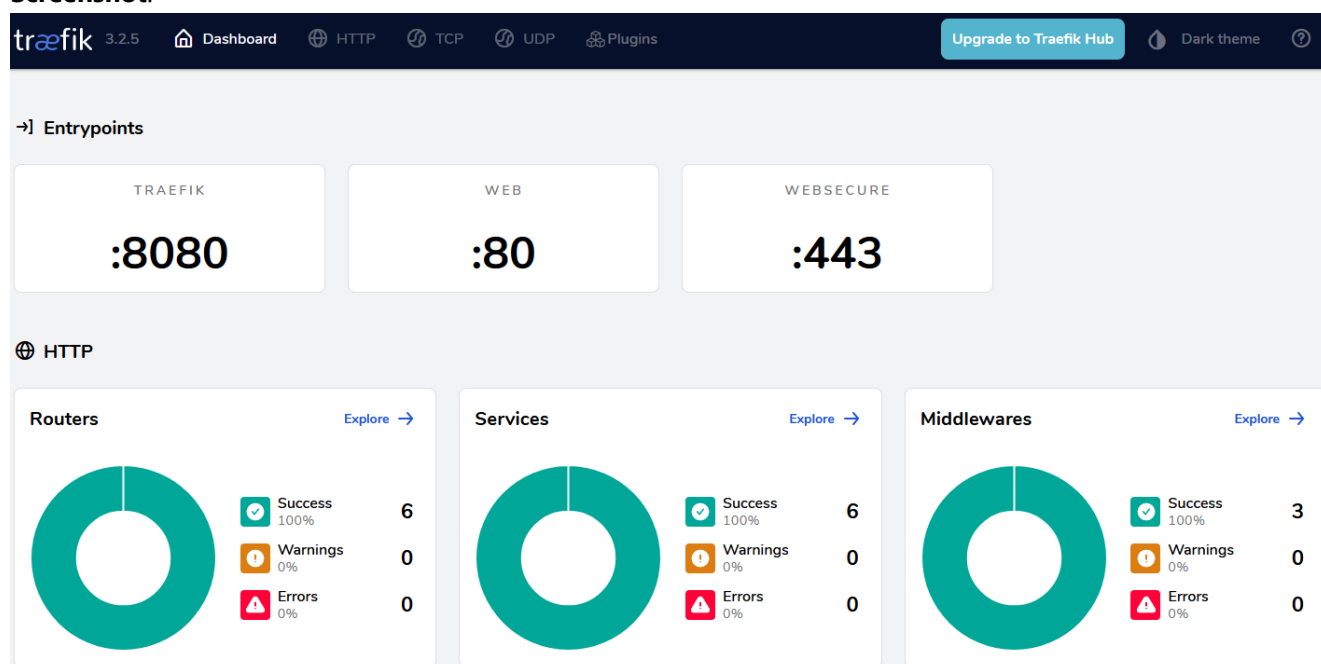
```
http://SERVER_IP:assigned_port
http://SERVER_IP:assigned_port
http://SERVER_IP:assigned_port
```

Note: Ports 8085-8095 are available for up to 11 replicas.

Expected Result:

- Page displays "David Maat has reached Milestone 1!!"
- Shows container hostname (different for each port)

Screenshot:



3. Test Load Balancing

Access the domain

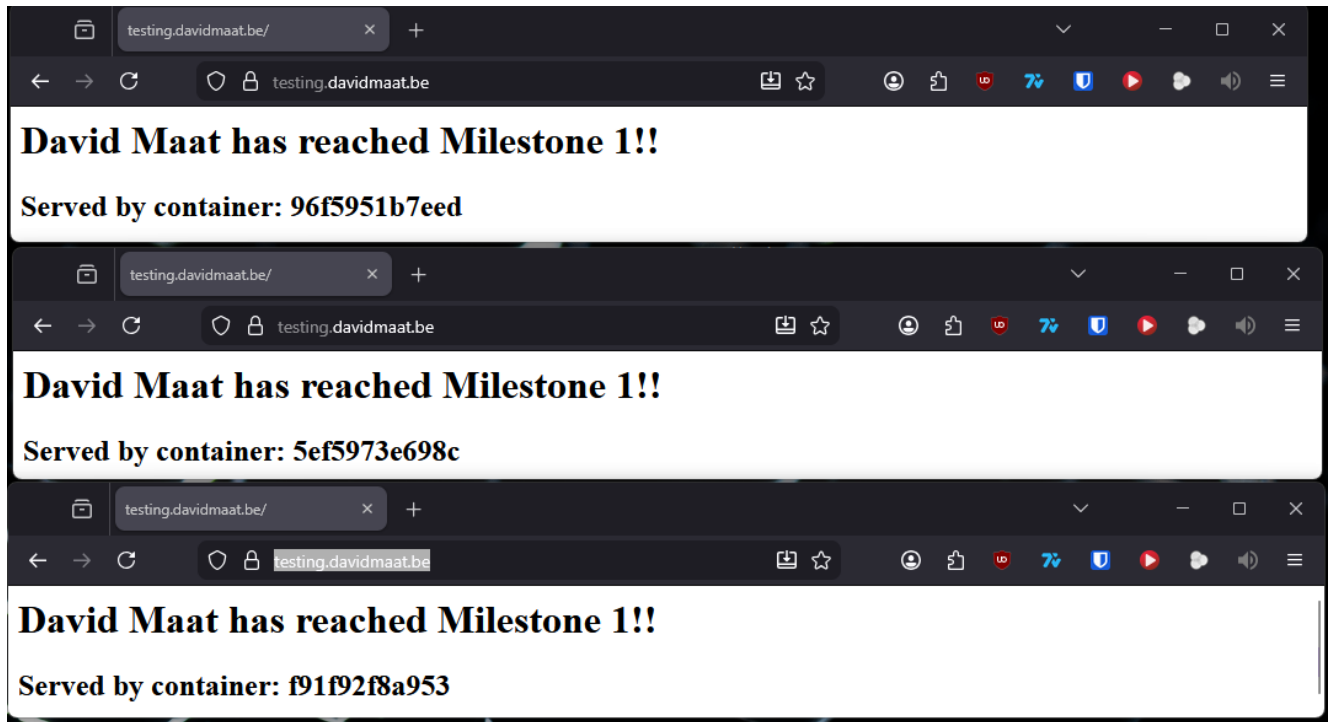
Refresh the page multiple times and observe the "Served by container" hostname changing between:

- contapa2-m1-dm-1
- contapa2-m1-dm-2

- `contapa2-m1-dm-3`

Explanation: Traefik distributes requests across all healthy backend servers.

Screenshot:



4. Access phpMyAdmin

Navigate to:

```
http://SERVER_IP:8081
```

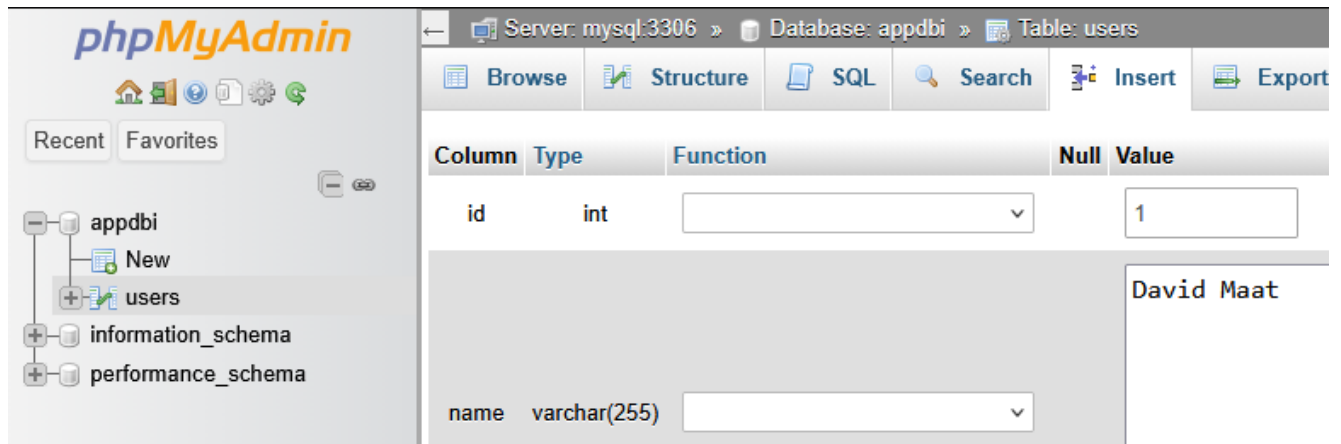
Login credentials:

- **Server:** `contsql-m1-dm`
- **Username:** Value from `MYSQL_USER` in `.env`
- **Password:** Value from `MYSQL_PASSWORD` in `.env`

What to check:

- Database `milestone_db` exists
- Table `users` contains "David Maat"

Screenshot:



Additional Resources

Docker Documentation

- [Docker Compose Reference](#)
- [Docker Networking](#)
- [Docker Volumes](#)

Traefik Documentation

- [Traefik Official Docs](#)
- [Let's Encrypt Integration](#)
- [Docker Provider](#)

MySQL Documentation

- [MySQL Docker Image](#)
- [MySQL Configuration](#)

PHP Documentation

- [PHP-FPM Configuration](#)
- [PHP MySQL Extension](#)

Conclusion

This Docker Compose stack demonstrates a production-ready architecture with:

- Load balancing across multiple web servers
- Automatic SSL certificate management
- Health monitoring and automatic recovery
- Persistent data storage
- Service isolation and dependency management
- Scalability (easily adjust replica count)

For questions or issues, refer to the troubleshooting section or consult the official documentation linked above.

Document Version: 1.0

Last Updated: October 26, 2025

Author: David Maat

Project: Linux Web Services - Milestone 1

Link to proof

<https://youtu.be/K4Ahl6TJ738>