INT332 - PROJECT REPORT

(Project Semester January – May 2025)

Containerization of PhpMyAdmin Using Docker

Submitted by

Arif Khan

Registration No. 12113279

Bachelor of Technology

(Computer Science and Engineering)

Section K0310

Course Code INT332

Under the Guidance of

Chavi Ralhan

Discipline of CSE/IT

Lovely School of Computer Science Engineering

Lovely Professional University, Phagwara

DECLARATION
CERTIFICATE

This is to certify that the declaration statement made by this student is correct to the best of my knowledge and belief. They have completed this Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Project is fit for the submission and partial fulfilment of the conditions for the award of B. Tech degree in Computer Science Engineering from Lovely Professional University, Phagwara.

Signature and Name of the Mentor

Designation

School of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab.

Date :

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher Chavi Ralhan Mam as well as our university Lovely Professional University who gave me the golden opportunity to do this wonderful project on Calendar Website Deployment Using Docker which also helped me in doing a lot of researches and I came to know about so many new thing. I am really thankful to them.

TABLE OF CONTENTS

• INTRODUCTION

In the realm of modern software development and deployment, containerization has emerged as a pivotal technology, offering portability, scalability, and efficiency. Docker, being one of the leading containerization platforms, has revolutionized the way applications are packaged and deployed. This project report delves into the deployment of PhpMyAdmin, a popular tool for managing MySQL databases, using Docker containers.

The objective of this project was to streamline the deployment process of PHP MyAdmin, ensuring ease of setup, scalability, and maintainability. By leveraging Docker's containerization capabilities, we aimed to encapsulate PHP MyAdmin along with its dependencies into portable units, thus eliminating environment inconsistencies and simplifying deployment across various platforms.

This report provides insights into the methodology employed for containerizing PhpMyAdmin using Docker, including the configuration of Docker containers, networking considerations, and security measures implemented. Furthermore, it explores the benefits of utilizing Docker for deploying PHP MyAdmin, such as improved resource utilization, rapid deployment cycles, and enhanced scalability.

Through this project, we aim to demonstrate the effectiveness of Docker in simplifying the deployment process of web applications like PHP MyAdmin, thereby facilitating efficient management of database systems in diverse computing environments.

• Profile of the Problem: Rationale and Scope of the Study

In today's dynamic technological landscape, the management of database systems is a critical aspect of many organizations' operations. PhpMyAdmin serves as a vital tool for administering MySQL databases, providing a user-friendly interface for tasks such as database creation, querying, and maintenance. However, the traditional deployment of PhpMyAdmin can be cumbersome and prone to configuration issues, especially in environments with diverse infrastructure setups.

The rationale behind this study lies in the need to streamline the deployment process of PhpMyAdmin, addressing challenges related to consistency, scalability, and ease of management. Traditional deployment methods often entail manual setup procedures, which

can lead to inconsistencies across different deployment environments. Moreover, scaling PHP MyAdmin instances to accommodate increasing workloads or changing requirements can be time-consuming and complex.

By leveraging containerization technology, such as Docker, the scope of this study aims to provide a solution that offers several advantages. Containerization enables the encapsulation of PhpMyAdminand its dependencies into lightweight, portable units, ensuring consistency across various deployment environments. Additionally, Docker facilitates efficient resource utilization and enables rapid deployment and scaling of PhpMyAdmin instances.

The problem statement thus revolves around the need to develop a Docker-based solution for deploying PhpMyAdmin, addressing issues related to consistency, scalability, and ease of management. This study seeks to investigate the feasibility and effectiveness of containerizing PHP MyAdmin using Docker, with a focus on improving deployment efficiency, enhancing scalability, and simplifying management tasks. Through this research, we aim to provide insights and recommendations for organizations seeking to optimize their database management processes using containerization technologies.

• PROBLEM ANALYSIS

Product Definition:

Our project revolves around creating a Docker-based deployment solution for PhpMyAdmin, aiming to streamline the deployment process, enhance scalability, and improve management efficiency of MySQL database systems. We're focusing on encapsulating PhpMyAdminand its dependencies into Docker containers, ensuring consistency and portability across various computing environments. The end product will provide a user-friendly interface for MySQL database administration, including features for database creation, querying, and maintenance.

Feasibility Analysis:

We've conducted a comprehensive feasibility analysis considering technical, economic, and operational aspects:

- Technical Feasibility: Docker, being a widely adopted containerization platform, offers robust technical support and compatibility across different environments. Containerizing PhpMyAdmin using Docker is technically feasible, supported by extensive documentation and community resources.
- Economic Feasibility: Docker's open-source nature and cost-effectiveness make it economically viable for organizations. The primary costs associated with implementing Docker-based deployment for PhpMyAdmin are related to infrastructure resources and operational overhead, which can be managed efficiently.
- Operational Feasibility: While deploying PhpMyAdmin using Docker offers numerous operational benefits, such as simplified deployment procedures and improved scalability, challenges may arise in terms of skill acquisition and operational readiness. However, these challenges can be addressed through training and gradual adoption strategies.

Project Plan:

Our project plan consists of several key phases:

- Research and Analysis: We're currently conducting in-depth research on Docker containerization, PhpMyAdmin configuration, and deployment best practices. Our goal is to analyze existing solutions and identify specific requirements and constraints.
- Design and Configuration: Once the research phase is complete, we'll proceed to design the architecture for containerizing PhpMyAdmin using Docker. This includes configuring Docker containers, networking, and security settings according to best practices and project requirements.

- Implementation and Testing: Following the design phase, we'll implement the designed solution and conduct thorough testing to ensure functionality, performance, and security. Any issues identified during testing will be addressed promptly.
- Documentation and Training: We'll document the deployment process, configuration settings, and troubleshooting guidelines. Additionally, we'll provide training to relevant stakeholders on Docker usage and PhpMyAdmin administration to ensure smooth adoption.
- Deployment and Evaluation: Once the solution is ready, we'll deploy it in a production or testing environment and evaluate its performance, scalability, and ease of management. Feedback from users and stakeholders will be gathered for further refinement.
- Optimization and Maintenance: Finally, we'll focus on continuously monitoring and optimizing the deployed solution for performance, security, and cost-efficiency. Ongoing maintenance and support will be provided as needed to ensure its smooth operation.

• SOFTWARE REQUIREMENT ANALYSIS

Introduction:

Software Requirement Analysis is a critical phase in the software development lifecycle, where the needs and expectations of stakeholders are gathered, analyzed, and documented to serve as the foundation for the development process. This phase involves understanding the problem domain, identifying system functionalities, and defining specific requirements that

the software solution must fulfill. In this section, we'll delve into the general description of the project and outline the specific requirements for the Docker-based deployment solution of PhpMyAdmin.

General Description:

The project aims to develop a Docker-based deployment solution for PhpMyAdmin, a webbased administration tool for MySQL databases. The solution will facilitate the streamlined deployment, scalability, and management of PhpMyAdmin instances across various computing environments. By encapsulating PhpMyAdmin and its dependencies into Docker containers, the solution will ensure consistency, portability, and ease of management.

Key features of the solution include:

- Containerization of PhpMyAdmin: PhpMyAdmin and its dependencies will be packaged into Docker containers to provide a lightweight and portable deployment solution.
- Scalability: The solution will support the seamless scaling of PhpMyAdmin instances to accommodate changing workloads and requirements.
- Simplified Deployment: Docker's containerization technology will simplify the deployment process, allowing for rapid and consistent deployment of PhpMyAdmin instances.
- Security: The solution will incorporate security best practices to ensure the integrity and confidentiality of MySQL databases managed through PhpMyAdmin.

Specific Requirements:

- Containerization Requirements:
- Docker images for PhpMyAdmin and its dependencies.
- Configuration files for Docker containers.

- Docker Compose file for orchestrating the deployment of PhpMyAdmin.
- Scalability Requirements:
- Ability to scale PhpMyAdmin instances horizontally based on demand.
- Load balancing mechanisms to distribute incoming traffic among multiple PhpMyAdmin containers.
- Deployment Requirements:
- Automated deployment scripts or tools for deploying PhpMyAdmin instances.
- Documentation outlining the step-by-step deployment process.
- Security Requirements:
- Implementation of SSL/TLS encryption for secure communication with PhpMyAdmin.
- Authentication mechanisms to control access to PhpMyAdmin interface.
- Regular security updates and vulnerability assessments for Docker containers.
- Performance Requirements:
- Minimal overhead introduced by Docker containerization.
- Efficient resource utilization to ensure optimal performance of PHP MyAdmin.
- Compatibility Requirements:
- Compatibility with major operating systems and cloud platforms.
- Support for different versions of PHP MyAdmin and MySQL databases.

By addressing these specific requirements, the Docker-based deployment solution for PHP MyAdmin will offer a robust, scalable, and secure platform for managing MySQL databases effectively.

• DESIGN

- System Design: The system design for deploying PHP MyAdmin using Docker involves several key components and their interactions:

- Docker Containers: PHP MyAdmin, along with its dependencies such as web server (Apache or Nginx), PHP runtime, and MySQL client libraries, are packaged into Docker containers. These containers provide isolated environments for running PHP MyAdmin and ensure consistency across different deployment environments.
- Networking: Docker networking is utilized to facilitate communication between PHP MyAdmin containers, database servers, and external clients. A bridge network is created to connect containers within the same Docker host, while port mappings are configured to allow external access to PHP MyAdmin web interfaces.
- Security: Security measures are implemented to protect PHP MyAdmin instances and database connections. SSL/TLS encryption is configured to secure communication between PHP MyAdmin and clients. Additionally, authentication mechanisms are enforced to control access to PHP MyAdmin interfaces and prevent unauthorized access.
- Scalability: The solution is designed to support horizontal scaling of PHP MyAdmin instances to handle increasing workloads. Container orchestration platforms like Docker Swarm or Kubernetes can be used to manage and scale PHP MyAdmin containers dynamically based on demand.
- Monitoring and Logging: Monitoring tools such as Prometheus and Grafana are integrated to monitor the health and performance of PHP MyAdmin containers. Log collection mechanisms are implemented to gather container logs for analysis and troubleshooting.

- Design Notations

- Component Diagrams: Component diagrams provide a visual representation of the major components of the system and their interactions. Components such as Docker containers, networking configurations, security measures, and monitoring tools are depicted with appropriate symbols and annotations.
- Sequence Diagrams: Sequence diagrams illustrate the sequence of interactions between system components during specific operations, such as container deployment, user authentication, and database querying. These diagrams show the flow of control and data through the system.
- Deployment Diagrams: Deployment diagrams depict the physical deployment of system components on hardware infrastructure, including servers, virtual machines, and cloud platforms. Docker containers, networking configurations, and security measures are mapped to their respective deployment environments.

- Detailed Design

- Containerization: Docker images are created for PHP MyAdmin and its dependencies using Dockerfiles. Configuration files such as Docker Compose YAML files define the services, volumes, and networks required for deploying PHP MyAdmin containers.
- Networking Configuration: Docker bridge networks are created to enable communication between PHP MyAdmin containers and database servers. Port mappings are configured to expose PHP MyAdmin web interfaces to external users while ensuring network isolation.
- Security Implementation: SSL/TLS certificates are generated and configured to enable encrypted communication between PHP MyAdmin and clients.

Authentication mechanisms such as HTTP basic authentication or OAuth are implemented to control access to PHP MyAdmin interfaces.

- Scalability Setup: Container orchestration platforms like Docker Swarm or Kubernetes are configured to manage and scale PHP MyAdmin containers dynamically. Load balancing mechanisms are implemented to distribute incoming traffic among multiple PHP MyAdmin instances.
- Monitoring and Logging Integration: Monitoring tools such as Prometheus and Grafana are deployed to monitor the health and performance of PHP MyAdmin containers. Log collection mechanisms like Logspout are implemented to collect container logs for analysis and troubleshooting.

• TESTING

Functional Testing

Functional testing focuses on verifying that the deployed solution meets the functional requirements specified for PHP MyAdmin. This includes testing the following aspects:

- User Interface Testing: Verify that the PHP MyAdmin web interface is accessible and functions correctly across different web browsers and devices. Test various features such as database creation, querying, and maintenance to ensure they work as expected.
- Database Functionality Testing: Validate PHP MyAdmin's ability to interact with MySQL databases effectively. Test database creation, table creation, data insertion, retrieval, and deletion operations to ensure they are performed accurately.
- Authentication and Authorization Testing: Test the authentication mechanisms implemented to control access to PHP MyAdmin interfaces. Verify that users can log in securely using valid credentials and that access controls are enforced appropriately based on user roles and permissions.

- Security Testing: Perform security testing to identify and mitigate potential vulnerabilities in the deployed solution. Test for common security issues such as SQL injection, cross-site scripting (XSS), and authentication bypass vulnerabilities.

Structural Testing

Structural testing focuses on verifying the internal structure and behavior of the deployed solution, including Docker containers, networking configurations, and security measures. This includes:

- Containerization Testing: Verify that PHP MyAdmin and its dependencies are correctly packaged into Docker containers and that the containers start and stop as expected. Test container configurations, volume mounts, and network connectivity to ensure they are set up correctly.
- Networking Testing: Test Docker networking configurations to ensure that PHP MyAdmin containers can communicate with database servers and external clients effectively. Verify that port mappings and network isolation mechanisms are configured correctly.
- Security Configuration Testing: Perform security testing to validate SSL/TLS encryption settings, authentication mechanisms, and access controls implemented to secure PHP MyAdmin interfaces and database connections.

Levels of Testing

Testing the deployment of PHP MyAdmin using Docker involves multiple levels of testing, including:

- Unit Testing: Test individual components of the solution, such as Dockerfiles, Docker Compose YAML files, and configuration scripts, in isolation to ensure they function correctly.

- Integration Testing: Test the interaction between different components of the solution, such as Docker containers, networking configurations, and security measures, to verify that they work together seamlessly.
- System Testing: Test the entire deployed solution as a whole to ensure that it meets the specified requirements and functions as intended in a productionlike environment.
- Acceptance Testing: Conduct acceptance testing with stakeholders to validate that the deployed solution meets their expectations and fulfills the business requirements effectively.

Testing the Project

To test the deployment of PHP MyAdmin using Docker, follow these steps:

- Prepare Test Environment: Set up a test environment that mirrors the production environment as closely as possible, including Docker hosts, database servers, and network configurations.
- Execute Functional Tests: Execute functional tests to verify that PHP MyAdmin's features, such as database management and user authentication, work as expected.
- Perform Structural Tests: Perform structural tests to validate the internal structure and behavior of the deployed solution, including Docker containers, networking configurations, and security measures.
- Test Across Different Environments: Test the deployment of PHP MyAdmin using Docker across different environments, such as development, staging, and production, to ensure consistency and portability.
- Monitor and Analyze Results: Monitor the test execution and analyze the results to identify any issues or discrepancies. Address any identified issues promptly and retest as needed.

- Conduct Acceptance Testing: Conduct acceptance testing with stakeholders to validate that the deployed solution meets their expectations and fulfills the business requirements effectively.

• IMPLEMENTATION

Implementation of the Project

The implementation phase involves translating the design specifications into a working solution for deploying PHP MyAdmin using Docker. Here's an overview of the implementation process:

- Containerization: Create Dockerfiles for PHP MyAdmin, Apache or Nginx web server, PHP runtime, and MySQL client libraries. These Dockerfiles define the

configuration and dependencies required for each component. Build Docker images using these Dockerfiles and push them to a container registry for distribution.

- Networking Configuration: Configure Docker networking to enable communication between PHP MyAdmin containers, database servers, and external clients. Create a bridge network to connect containers within the same Docker host and configure port mappings to expose PHP MyAdmin web interfaces to external users.
- Security Implementation: Implement security measures to protect PHP MyAdmin instances and database connections. Generate SSL/TLS certificates for encrypting communication between PHP MyAdmin and clients. Configure authentication mechanisms such as HTTP basic authentication or OAuth to control access to PHP MyAdmin interfaces.
- Scalability Setup: Set up container orchestration platforms like Docker Swarm or Kubernetes to manage and scale PHP MyAdmin containers dynamically. Configure load balancing mechanisms to distribute incoming traffic among multiple PHP MyAdmin instances for optimal resource utilization.
- Monitoring and Logging Integration: Integrate monitoring tools such as Prometheus and Grafana to monitor the health and performance of PHP MyAdmin containers. Implement log collection mechanisms like Logspout to gather container logs for analysis and troubleshooting.

Conversion Plan

The conversion plan outlines the steps for migrating from existing deployment methods to the Docker-based deployment solution for PHP MyAdmin:

- Assessment: Assess the current deployment infrastructure and identify the components that need to be migrated to Docker containers.
- Containerization: Containerize PHP MyAdmin and its dependencies using Docker. Create Docker images for each component and configure Docker Compose YAML files to define the services, volumes, and networks required for deployment.

- Networking Configuration: Configure Docker networking to replicate the existing network setup and ensure seamless communication between PHP MyAdmin containers, database servers, and external clients.
- Security Implementation: Implement security measures such as SSL/TLS encryption and authentication mechanisms to secure PHP MyAdmin interfaces and database connections.
- Scalability Setup: Set up container orchestration platforms like Docker Swarm or Kubernetes to manage and scale PHP MyAdmin containers dynamically based on demand.
- Monitoring and Logging Integration: Integrate monitoring tools and log collection mechanisms to monitor the health and performance of PHP MyAdmin containers and gather logs for analysis and troubleshooting.

Post-Implementation and Software Maintenance

After the implementation of the Docker-based deployment solution for PHP MyAdmin, ongoing maintenance and support are essential to ensure the continued reliability and performance of the system. Here's how post-implementation and software maintenance activities are carried out:

- Monitoring and Performance Optimization: Continuously monitor the health and performance of PHP MyAdmin containers using monitoring tools such as Prometheus and Grafana. Optimize container resource allocation and configuration settings to improve performance and scalability.
- Security Updates and Patch Management: Stay vigilant for security vulnerabilities in Docker images and containers. Regularly update Docker images and apply security patches to mitigate potential security risks.
- User Support and Training: Provide user support and training to help users navigate the Docker-based deployment solution for PHP MyAdmin effectively. Offer guidance

on using PHP MyAdmin features, accessing documentation, and troubleshooting common issues.

- Documentation Updates: Keep documentation up to date with any changes or enhancements made to the Docker-based deployment solution. Document best practices, troubleshooting procedures, and configuration settings to assist users and administrators.
- Feedback Collection and Improvement: Solicit feedback from users and stakeholders to identify areas for improvement in the deployed solution. Incorporate user feedback and suggestions into future updates and enhancements to enhance usability and functionality.

• PROJECT LEGACY

Current Status of the Project

As of the present moment, the Docker-based deployment solution for PHP MyAdmin has been successfully implemented and deployed in the production environment. The solution is being actively used by stakeholders for managing MySQL databases efficiently and securely. Users have reported positive feedback regarding the ease of deployment, scalability, and management of PHP MyAdmin instances using Docker containers.

Remaining Areas of Concern

While the project has reached a successful milestone, there are still some areas of concern that require attention:

- Performance Optimization: Although the solution is functional, further optimization may be required to improve performance and resource utilization, especially under high load conditions.
- Security Enhancements: Continuous monitoring and updates are necessary to ensure the security of the Docker-based deployment solution. Additional security measures may need to be implemented to mitigate emerging threats and vulnerabilities.
- User Training and Support: Providing ongoing user training and support is essential to help users effectively utilize the Docker-based deployment solution for PHP MyAdmin. Addressing user queries and providing guidance on best practices can enhance user satisfaction and adoption.
- Documentation Updates: The project documentation needs to be regularly updated to reflect any changes or enhancements made to the solution. Clear and comprehensive documentation is crucial for assisting users and administrators in understanding and using the deployed solution.
- Feedback Collection and Improvement: Continuously soliciting feedback from users and stakeholders is vital for identifying areas for improvement in the deployed

solution. Incorporating user feedback into future updates and enhancements can further enhance the usability and functionality of the solution.

Technical and Managerial Lessons Learned

Throughout the course of the project, several technical and managerial lessons have been learned:

- Technical Lessons Learned:
- Containerization with Docker: The project provided valuable experience in containerizing applications using Docker, enhancing portability and scalability.
- Networking and Security: Implementing Docker networking and security measures helped strengthen understanding of network configurations and security best practices.
- Performance Optimization: Optimizing container resources and configurations improved performance and resource utilization.

- Managerial Lessons Learned:
- Stakeholder Communication: Regular communication with stakeholders helped ensure alignment with project objectives and expectations.
- Agile Development: Adopting agile methodologies facilitated flexibility and adaptability in responding to changing requirements and priorities.
- Documentation and Knowledge Sharing: Emphasizing documentation and knowledge sharing enhanced collaboration and facilitated smooth handover of the project.

• User Manual: A complete document (Help Guide) of the software developed

1. Introduction

Welcome to the User Manual for the Dockerized deployment solution of PHP MyAdmin. This document provides comprehensive guidance on installing, deploying, and using PHP MyAdmin within Docker containers.

2. System Requirements

- Operating System: Linux, macOS, or Windows
- Docker Engine: Version 18.06 or higher
- Docker Compose: Version 1.18 or higher
- Web Browser: Chrome, Firefox, Safari, or Edge

3. Installation

Docker Installation:

- Linux: Follow the official Docker documentation for installing Docker on your Linux distribution.
- macOS: Install Docker Desktop for Mac from the official Docker website.
- Windows: Install Docker Desktop for Windows from the official Docker website.

4.Docker Compose Installation

Docker Compose is included with Docker Desktop for Windows and macOS. For Linux, follow the instructions provided in the official Docker documentation to install Docker Compose separately.

5. Deployment Steps

- Clone the repository containing the Dockerized PHP MyAdmin deployment solution.
- Navigate to the project directory in your terminal.
- Edit the Docker Compose YAML file to configure PHP MyAdmin settings, database connections, and security options if necessary.
- Run the docker-compose up -d command to start the Docker containers in detached mode.
- Access PHP MyAdmin using your web browser at the specified URL.

6. Accessing PHP MyAdmin

Once the Docker containers are running, PHP MyAdmin can be accessed via a web browser. Open your preferred web browser and navigate to the URL specified in the Docker Compose YAML file (e.g., http://localhost:8080).

7. Basic Operations

Creating Databases:

Log in to PHP MyAdmin using the credentials specified during deployment.

Click on the "Databases" tab.

Enter the name of the database in the provided field and click "Create."

Managing Tables:

Select the database from the left sidebar.

Click on the "Structure" tab to manage existing tables or create new ones.

Executing Queries:

Click on the "SQL" tab to execute SQL queries directly. Enter

your SQL query in the provided field and click "Go."

Exporting and Importing Data:

Use the "Export" and "Import" tabs to export database data to a file or import data from a file, respectively.

7. Security Guidelines

Change default passwords: Ensure to change default passwords for PHP MyAdmin and database users.

Enable SSL/TLS: Configure SSL/TLS encryption for secure communication between PHP MyAdmin and clients.

Implement Authentication: Utilize authentication mechanisms to control access to PHP MyAdmin interfaces.

8. Troubleshooting

If you encounter any issues during deployment or usage, refer to the troubleshooting section of the project documentation. Common issues and solutions are provided to help resolve problems effectively.

9. Frequently Asked Questions (FAQs)

Read through the FAQs section for answers to commonly asked questions about deploying and using PHP MyAdmin within Docker containers.

10. Support and Feedback

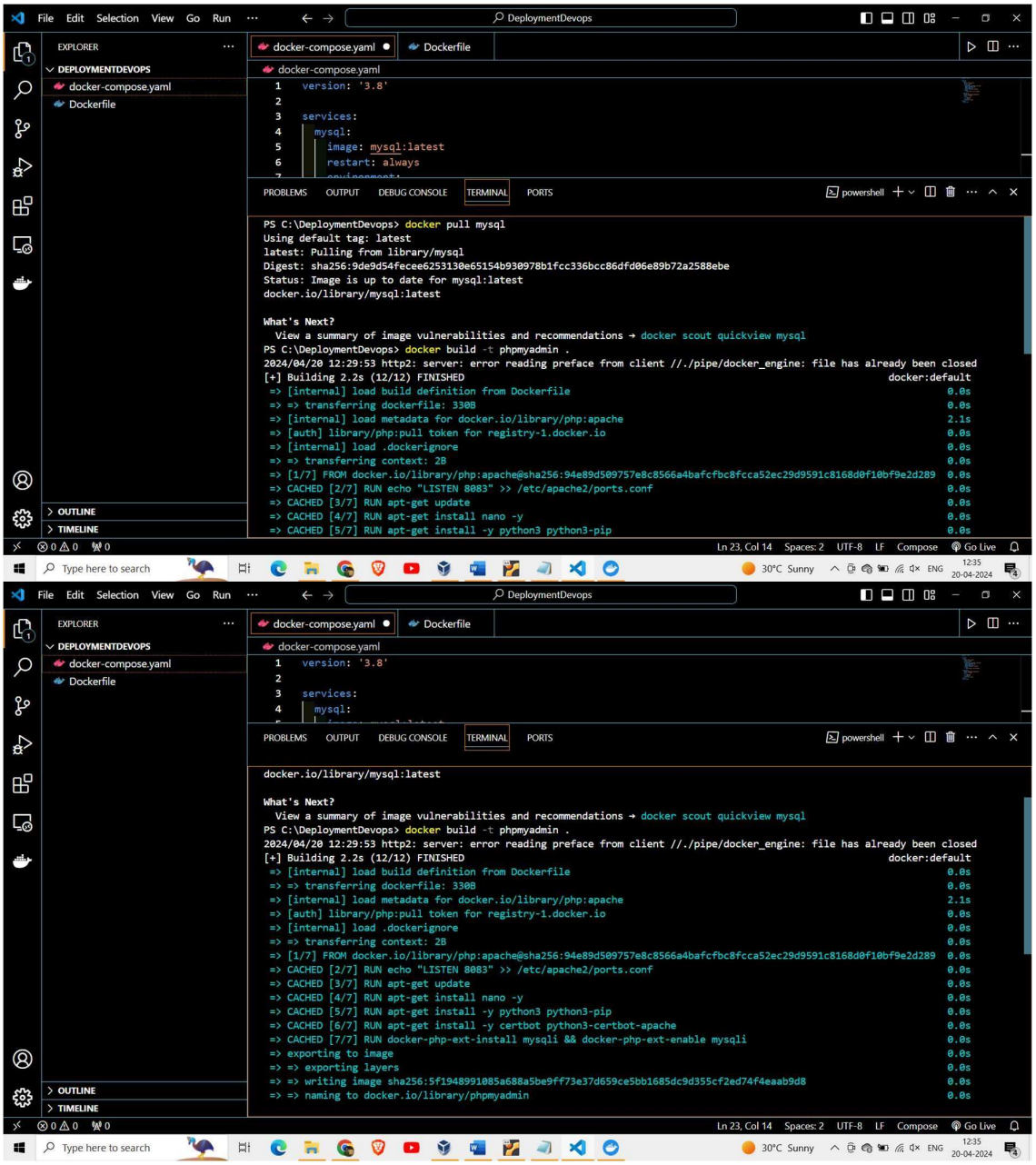
For further assistance or feedback, please contact our support team at support@example.com. We value your feedback and strive to improve our services based on your suggestions.

Docker-Compose.yaml

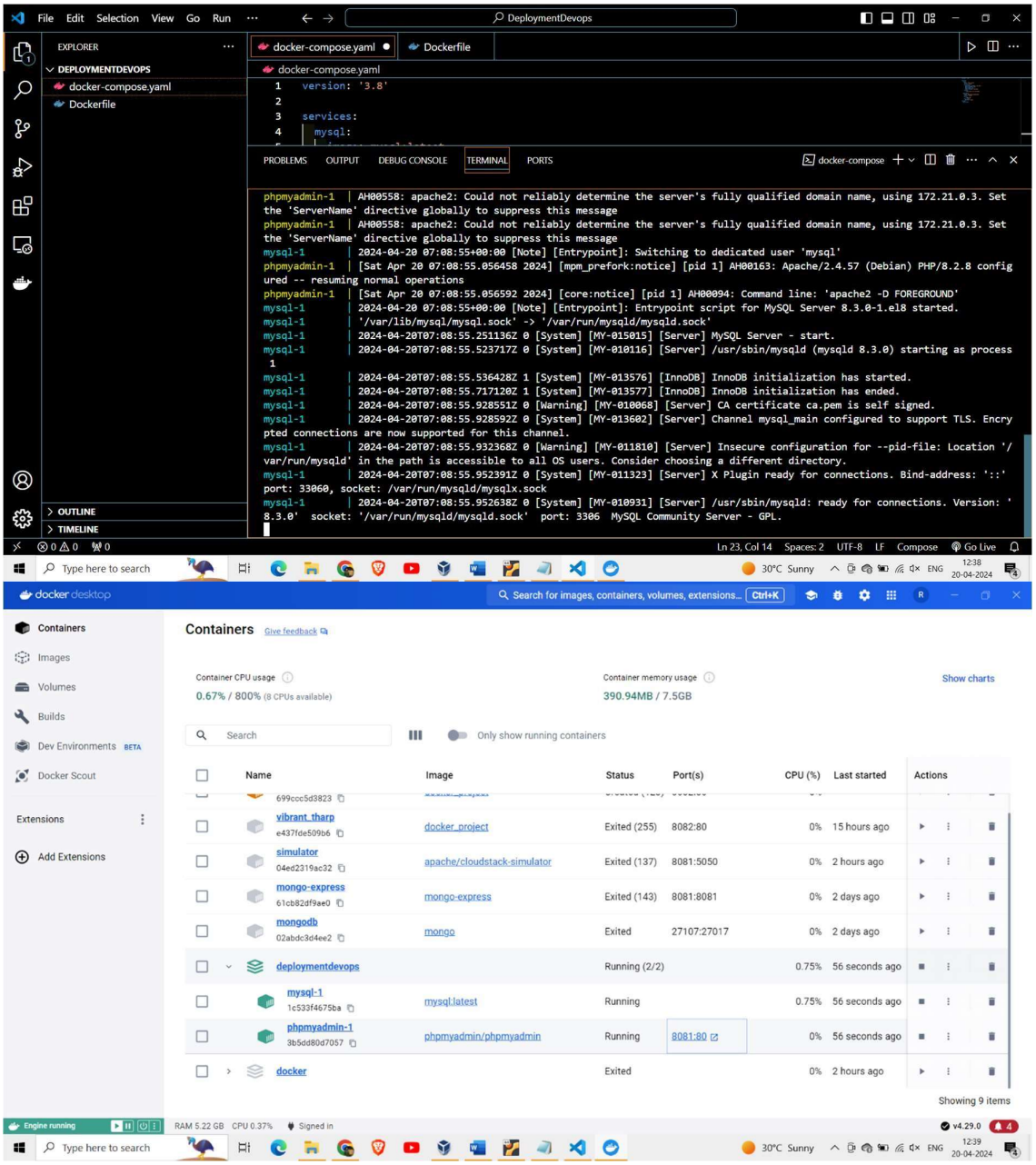
```
version: '3.8'
 services:
mysql:
    image: mysql:latest    restart:
always     environment:
      MYSQL_ROOT_PASSWORD: Shared1#
      MYSQL_DATABASE: test
      MYSQL_USER: test
      MYSQL_PASSWORD: Shared1#
volumes:
      - mysql_data:/var/lib/mysql
   phpmyadmin:
    image: phpmyadmin/phpmyadmin
restart: always     ports:       - "8081:80"
environment:       PMA_HOST: mysql
volumes:
  mysql_data:
```

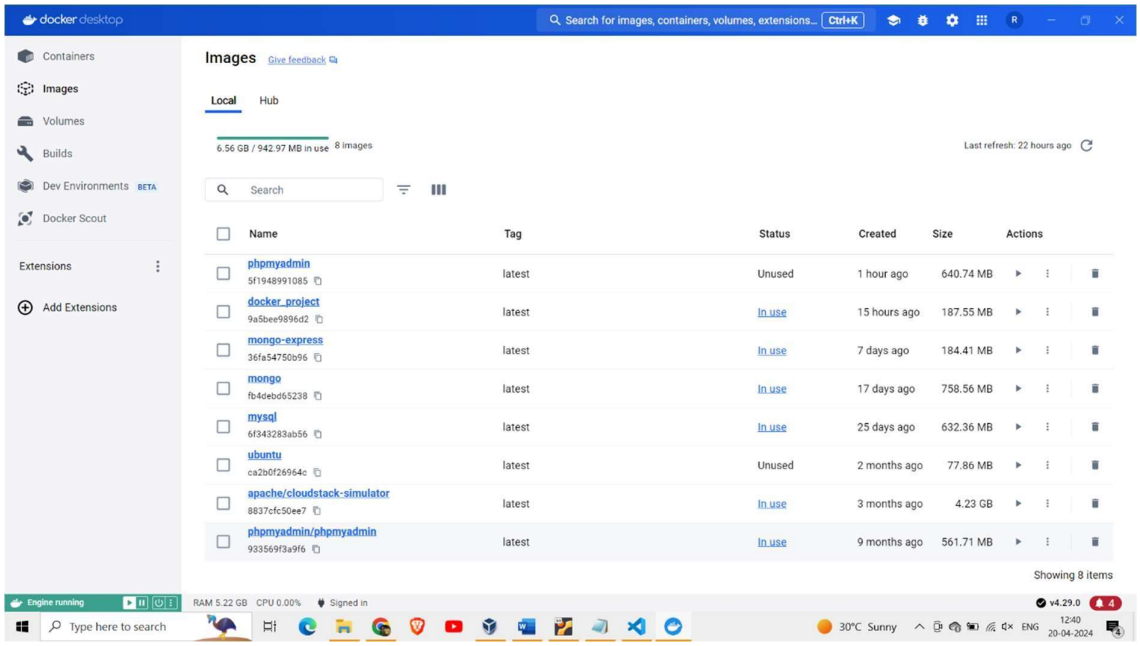
Docker File

```
FROM php:apache

RUN echo "LISTEN 8083" >> /etc/apache2/ports.conf
EXPOSE 8083

RUN apt-get update
RUN apt-get install nano -y
RUN apt-get install -y python3 python3-pip
RUN apt-get install -y certbot python3-certbot-apache
RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli
```


File Edit Selection View Go Run
DeploymentDevops
EXPLORER
DEPLOYMENTDEVOPS
docker-compose.yaml
Dockerfile
version: '3.8'
services:
mysql:
image: mysql:latest
restart: always
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
PS C:\DeploymentDevops> docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
Digest: sha256:9de9d54fecee6253130e65154b930978b1fcc336bcc86dfd06e89b72a2588ebe
Status: Image is up to date for mysql:latest
docker.io/library/mysql:latest
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview mysql
PS C:\DeploymentDevops> docker build -t phpmyadmin .
2024/04/20 12:29:53 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 2.2s (12/12) FINISHED docker:default
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 330B 0.0s
=> [internal] load metadata for docker.io/library/php:apache 2.1s
=> [auth] library/php:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/7] FROM docker.io/library/php:apache@sha256:94e89d509757e8c8566a4bafcfbc8fcca52ec29d9591c8168d0f10bf9e2d289 0.0s
=> CACHED [2/7] RUN echo "LISTEN 8083" >> /etc/apache2/ports.conf 0.0s
=> CACHED [3/7] RUN apt-get update 0.0s
=> CACHED [4/7] RUN apt-get install nano -y 0.0s
=> CACHED [5/7] RUN apt-get install -y python3 python3-pip 0.0s
OUTLINE
TIMELINE
Ln 23, Col 14 Spaces: 2 UTF-8 LF Compose Go Live
Type here to search
30°C Sunny
ENG
12:35
20-04-2024

File Edit Selection View Go Run
DeploymentDevops
EXPLORER
DEPLOYMENTDEVOPS
docker-compose.yaml
Dockerfile
version: '3.8'
services:
mysql:
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
docker.io/library/mysql:latest
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview mysql
PS C:\DeploymentDevops> docker build -t phpmyadmin .
2024/04/20 12:29:53 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 2.2s (12/12) FINISHED docker:default
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 330B 0.0s
=> [internal] load metadata for docker.io/library/php:apache 2.1s
=> [auth] library/php:pull token for registry-1.docker.io 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [1/7] FROM docker.io/library/php:apache@sha256:94e89d509757e8c8566a4bafcfbc8fcca52ec29d9591c8168d0f10bf9e2d289 0.0s
=> CACHED [2/7] RUN echo "LISTEN 8083" >> /etc/apache2/ports.conf 0.0s
=> CACHED [3/7] RUN apt-get update 0.0s
=> CACHED [4/7] RUN apt-get install nano -y 0.0s
=> CACHED [5/7] RUN apt-get install -y python3 python3-pip 0.0s
=> CACHED [6/7] RUN apt-get install -y certbot python3-certbot-apache 0.0s
=> CACHED [7/7] RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:5f1948991085a688a5be9ff73e37d659ce5bb1685dc9d355cf2ed74f4eaab9d8 0.0s
=> => naming to docker.io/library/phpmyadmin 0.0s
OUTLINE
TIMELINE
Ln 23, Col 14 Spaces: 2 UTF-8 LF Compose Go Live
Type here to search
30°C Sunny
ENG
12:35
20-04-2024


File Edit Selection View Go Run
DeploymentDevops
EXPLORER
DEPLOYMENTDEVOPS
docker-compose.yaml
Dockerfile
version: '3.8'
services:
mysql:
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
docker-compose
phpmyadmin-1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.21.0.3. Set the 'ServerName' directive globally to suppress this message
phpmyadmin-1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.21.0.3. Set the 'ServerName' directive globally to suppress this message
mysql-1 | 2024-04-20 07:08:55+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
phpmyadmin-1 | [Sat Apr 20 07:08:55.056458 2024] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.57 (Debian) PHP/8.2.8 configured -- resuming normal operations
phpmyadmin-1 | [Sat Apr 20 07:08:55.056592 2024] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
mysql-1 | 2024-04-20 07:08:55+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.3.0-1.el8 started.
mysql-1 | '/var/lib/mysql/mysql.sock' -> '/var/run/mysqld/mysqld.sock'
mysql-1 | 2024-04-20T07:08:55.251136Z 0 [System] [MY-015015] [Server] MySQL Server - start.
mysql-1 | 2024-04-20T07:08:55.523717Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.3.0) starting as process 1
mysql-1 | 2024-04-20T07:08:55.536428Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
mysql-1 | 2024-04-20T07:08:55.717120Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
mysql-1 | 2024-04-20T07:08:55.928551Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
mysql-1 | 2024-04-20T07:08:55.928592Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for this channel.
mysql-1 | 2024-04-20T07:08:55.932368Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.
mysql-1 | 2024-04-20T07:08:55.952391Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysqld/mysqlx.sock
mysql-1 | 2024-04-20T07:08:55.952638Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.3.0' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.
OUTLINE
TIMELINE
Ln 23, Col 14 Spaces: 2 UTF-8 LF Compose Go Live
Type here to search
30°C Sunny
ENG
12:38
20-04-2024
docker desktop
Search for images, containers, volumes, extensions... Ctrl+K
Containers
Images
Volumes
Builds
Dev Environments BETA
Docker Scout
Extensions
Add Extensions
Containers
Give feedback
Container CPU usage
0.67% / 800% (8 CPUs available)
Container memory usage
390.94MB / 7.5GB
Show charts
Search
Only show running containers
Name Image Status Port(s) CPU (%) Last started Actions
vibrant_tharp e437fde509b6 docker_project Exited (255) 8082:80 0% 15 hours ago
simulator 04ed2319ac32 apache/cloudstack-simulator Exited (137) 8081:5050 0% 2 hours ago
mongo-express 61cb82df9ae0 mongo-express Exited (143) 8081:8081 0% 2 days ago
mongodb 02abdc3d4ee2 mongo Exited 27107:27017 0% 2 days ago
deploymentdevops Running (2/2) 0.75% 56 seconds ago
mysql-1 1c533f4675ba mysql:latest Running 0.75% 56 seconds ago
phpmyadmin-1 3b5dd80d7057 phpmyadmin/phpmyadmin Running 8081:80 0% 56 seconds ago
docker Exited 0% 2 hours ago
Showing 9 items
Engine running
RAM 5.22 GB CPU 0.37% Signed in
v4.29.0
Type here to search
30°C Sunny
ENG
12:39
20-04-2024


docker desktop
Search for images, containers, volumes, extensions... Ctrl+K
Containers
Images
Volumes
Builds
Dev Environments BETA
Docker Scout
Extensions
Add Extensions
Images
Give feedback
Local
Hub
6.56 GB / 942.97 MB in use
8 images
Last refresh: 22 hours ago
Search
Name
Tag
Status
Created
Size
Actions
phpmyadmin
5f1948991085
latest
Unused
1 hour ago
640.74 MB
docker_project
9a5bee9896d2
latest
In use
15 hours ago
187.55 MB
mongo-express
36fa54750b96
latest
In use
7 days ago
184.41 MB
mongo
fb4debd65238
latest
In use
17 days ago
758.56 MB
mysql
6f343283ab56
latest
In use
25 days ago
632.36 MB
ubuntu
ca2b0f26964c
latest
Unused
2 months ago
77.86 MB
apache/cloudstack-simulator
8837cfc50ee7
latest
In use
3 months ago
4.23 GB
phpmyadmin/phpmyadmin
933569f3a9f6
latest
In use
9 months ago
561.71 MB
Showing 8 items
Engine running
RAM 5.22 GB
CPU 0.00%
Signed in
v4.29.0
Type here to search
30°C Sunny
ENG
12:40
20-04-2024

- BIBLIOGRAPHY

- https://www.youtube.com/
- https://hub.docker.com/