
Pilotage de projet DevOps

M2 MIAGE

Université Toulouse III - Paul Sabatier

Rémi Saurel

Contents

1	Introduction	2
2	Architecture	2
2.1	Exercice VIII	2
2.2	Exercice IX	2
3	Conclusion	3
4	Annexes	3

1 Introduction

Ce projet de pilotage de projet DevOps a pour but de mettre en pratique les connaissances acquises lors du cours avec l'utilisation de Docker et la mise en place de plusieurs services. 2 exercices seront proposés dans ce rendu, à savoir les exercices VIII et IX. Toutes les justifications sont présentes dans la partie 4, mais vous pouvez retrouver toutes les sources dans **ce repository GitHub que j'ai créé**.

2 Architecture

2.1 Exercice VIII

Architecture générale

La figure 1 présente l'architecture générale de cet exercice. On y retrouve un fichier `docker-compose.yml` qui permet de lancer les 2 services : **app** et **db8**. Le service **app** est le projet Spring Boot fourni. Le service **db8** est une base de données MySQL.

Types de réseaux

Le réseau utilisé est le réseau **bridge** de Docker. Il permet de créer un réseau privé pour les conteneurs. Les ports 3306 et 8080 sont exposés pour pouvoir accéder à la base de données et à l'application Spring Boot.

Types de stockages

Le stockage utilisé est le stockage **volume** de Docker, ici nommé **db_data**. Il permet de créer un volume pour les conteneurs. Il est utilisé pour la base de données afin de pouvoir conserver les données même si le conteneur est supprimé.

2.2 Exercice IX

Architecture générale

La figure 4 présente l'architecture générale de cet exercice. On y retrouve un fichier `docker-compose.yml` qui permet de lancer les 4 services : **app** et **db**. Le service **app** est le projet Spring Boot fourni. Le service **db** est une base de données MySQL. Le service phpMyAdmin est un service permettant d'interagir avec la base de données. Le service **Sonarqube** est un service permettant d'analyser le code source.

Types de réseaux

Le réseau utilisé est le réseau **bridge** de Docker. Un réseau a été explicitement créé (le réseau 'exo9') dans le docker compose et permet aux différents services d'interagir entre eux. Les ports 3306, 8080 et 9000 sont exposés pour pouvoir accéder à la base de données, à l'application Spring Boot et à Sonarqube.

Types de stockages

Comme pour l'exercice VIII, le stockage utilisé est le stockage **volume** de Docker, ici nommé **db_data**. Il est utilisé pour la base de données afin de pouvoir conserver les données même si le conteneur est supprimé.

3 Conclusion

4 Annexes

Vous pouvez retrouver toutes les sources sur le repository GitHub que j'ai créé.

Exercice VIII

Commandes

Commandes à réaliser : Dans le fichier Dockerfile, commenter / décommenter les lignes pour build ou non l'application Spring Boot.

```
docker compose up -d
```

Pour arrêter et supprimer les containers, vous pouvez utiliser :

```
docker compose down
```

Résultats

Sources

Dockerfile :

```
# FIRST VERSION WITH BUILD
# FROM maven:3.9.5-eclipse-temurin-21 AS builder
#
# WORKDIR /app
#
# COPY pom.xml .
```

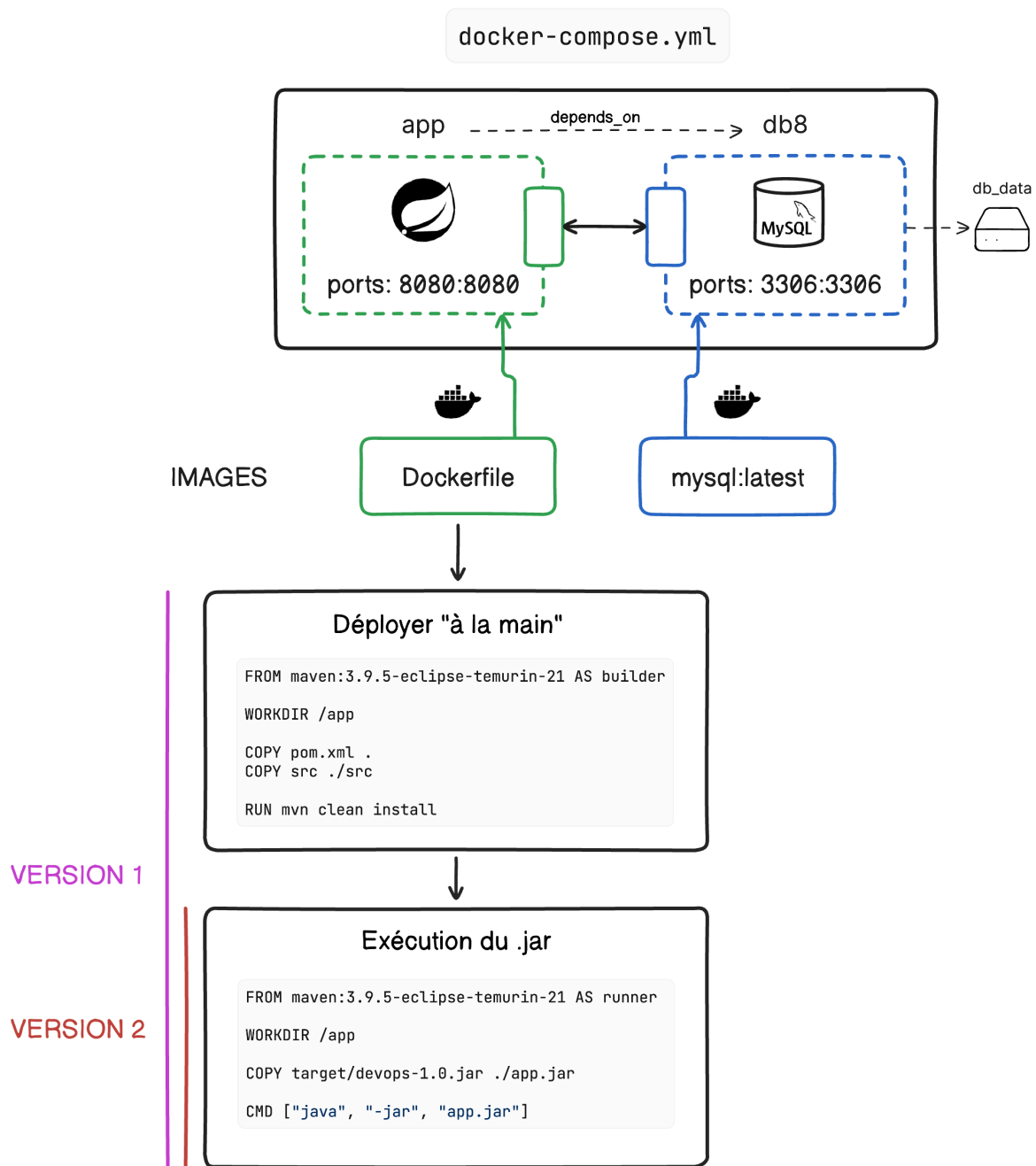


Figure 1: Architecture générale de l'exercice VIII

exo8	Running (2/2)	0.97%	16 seconds ago
app-1	Running	0.29%	16 seconds ago
db8	Running	0.68%	16 seconds ago

Figure 2: Les services lancés sur Docker Desktop

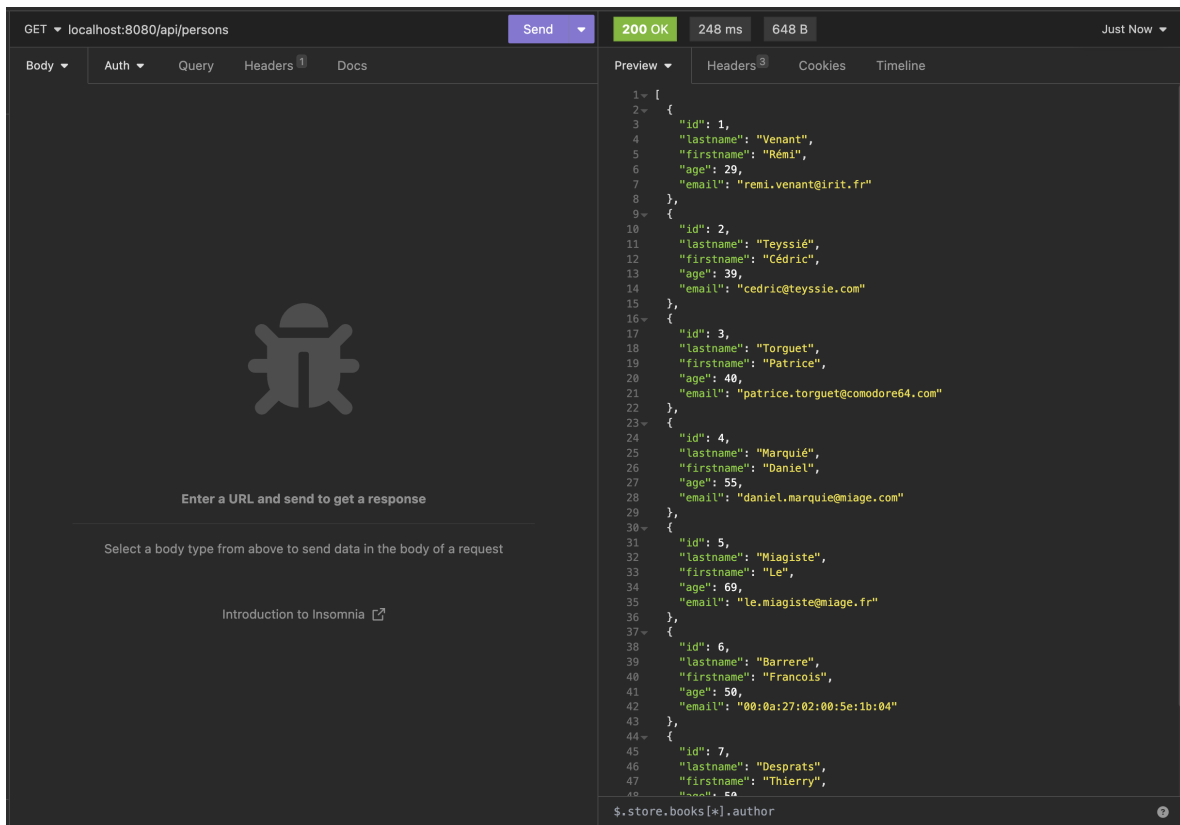


Figure 3: Test de l'app Spring Boot et de la BDD pour l'exercice 8

```

# COPY src ./src
#
# RUN mvn clean install
#
# FROM maven:3.9.5-eclipse-temurin-21 AS runner
#
# WORKDIR /app
#
# COPY --from=builder /app/target/*.jar ./app.jar
#
# ENTRYPOINT ["java", "-jar", "app.jar"]

```

```

# SECOND VERSION WITHOUT BUILD
FROM maven:3.9.5-eclipse-temurin-21 AS runner

```

```
WORKDIR /app
```

```
COPY target/devops-1.0.jar ./app.jar
```

```
CMD ["java", "-jar", "app.jar"]
```

docker-compose.yml :

```
version: '3'

services:
  db8:
    image: mysql:latest
    container_name: db8
    volumes:
      - db_data:/var/lib/mysql
      - ./data.sql:/docker-entrypoint-initdb.d/init.sql
    environment:
      MYSQL_USER: dev
      MYSQL_DATABASE: test
      MYSQL_ROOT_PASSWORD: root
      MYSQL_PASSWORD: root
    ports:
      - "3306:3306"

  app:
    build: .
    ports:
      - "8080:8080"
    depends_on:
      - db8

volumes:
  db_data:
```

Exercice IX

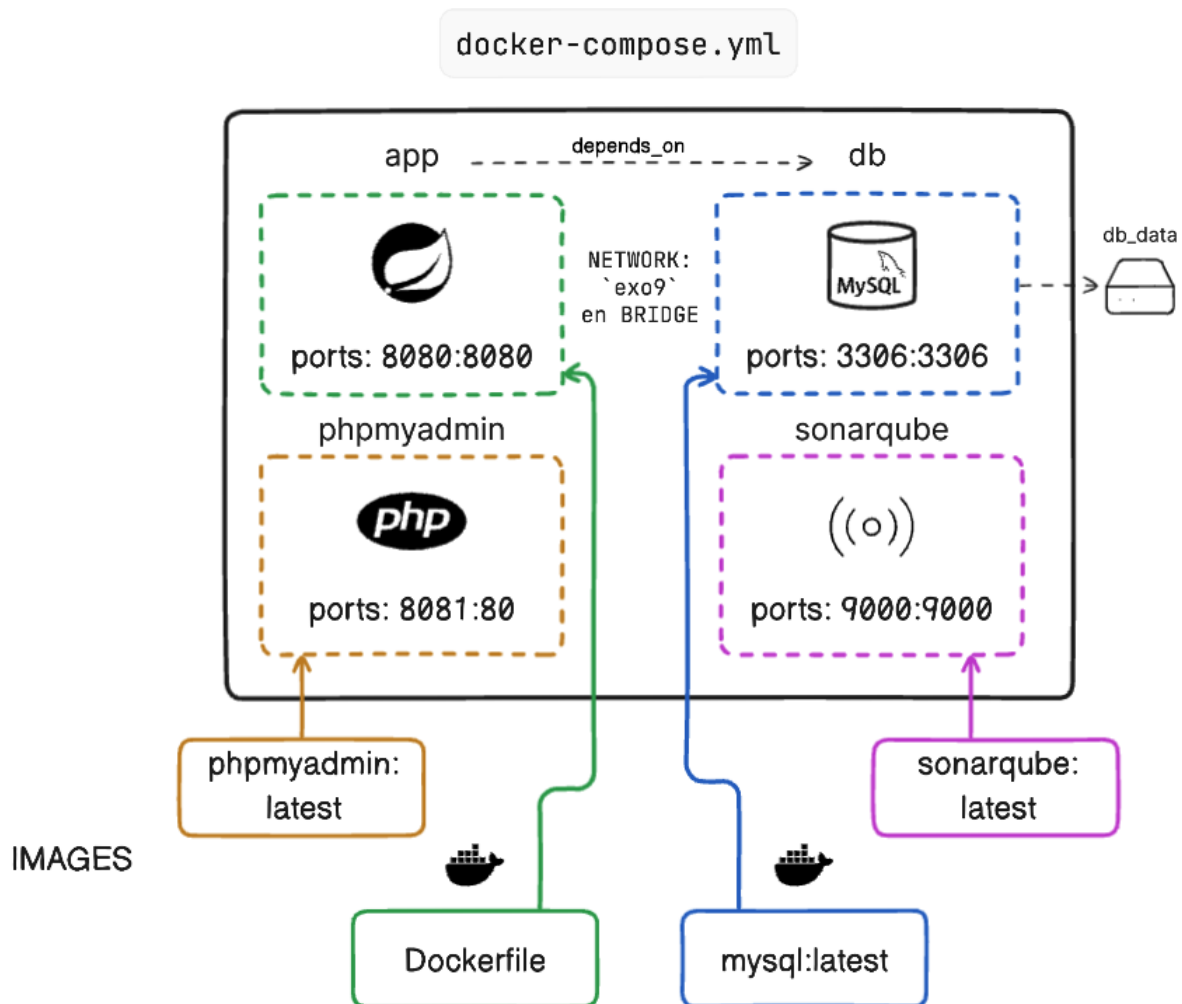


Figure 4: Architecture de l'exercice IX

Commandes

Commandes à réaliser :

```
docker compose up -d
```

Pour arrêter et supprimer les containers, vous pouvez utiliser :

```
docker compose down
```

Vous pouvez retrouver les différents services sur les liens suivants :

- <http://localhost:8080> pour l'application Spring Boot
- <http://localhost:8081> pour phpMyAdmin
- <http://localhost:9000> pour Sonarqube

Résultats

exo9	Running (4/4)	0%	1 second ago
app	Running	0%	1 second ago
phpmyadmin	Running	0%	1 second ago
sonarqube	Running	0%	1 second ago
db	Running	0%	1 second ago

Figure 5: Les services lancés sur Docker Desktop

The screenshot shows the phpMyAdmin interface for a database named 'test'. The left sidebar shows the database structure with 'test' selected. The main area displays the 'personnes' table structure. The table has two columns: 'personnes' and 'personnes_seq'. The 'personnes' column is of type 'InnoDB' and has a size of 16,0 kio. The 'personnes_seq' column is also of type 'InnoDB' and has a size of 16,0 kio. The table is named 'personnes' and is located in the 'test' database.

Table	Action	Lignes	Type	Interclassement	Taille	Pe
personnes	Parcourir Structure Rechercher Insérer Vider Supprimer	7	InnoDB	utf8mb3_general_ci	16,0 kio	
personnes_seq	Parcourir Structure Rechercher Insérer Vider Supprimer	1	InnoDB	utf8mb4_0900_ai_ci	16,0 kio	
2 tables	Somme	8	InnoDB	utf8mb4_0900_ai_ci	32,0 kio	

Figure 6: Interface phpMyAdmin avec la table 'personnes'

The screenshot shows the SonarQube interface. The top navigation bar includes 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and 'More'. The left sidebar shows 'My Favorites' and 'All' tabs. The main area displays the project 'exo9' with a status of 'Passed'. The project details show 'Last analysis: 20 hours ago - 121 Lines of Code - Java, XML'. The project has 0 Bugs, 0 Vulnerabilities, 0 Hotspots Reviewed, 1 Code Smells, 0.0% Coverage, and 0.0% Duplications.

Project	Status	Last analysis	Lines of Code	Language	Bugs	Vulnerabilities	Hotspots Reviewed	Code Smells	Coverage	Duplications
exo9	PUBLIC	20 hours ago	121	Java, XML	0	0	0	1	0.0%	0.0%

Figure 7: Interface Sonarqube

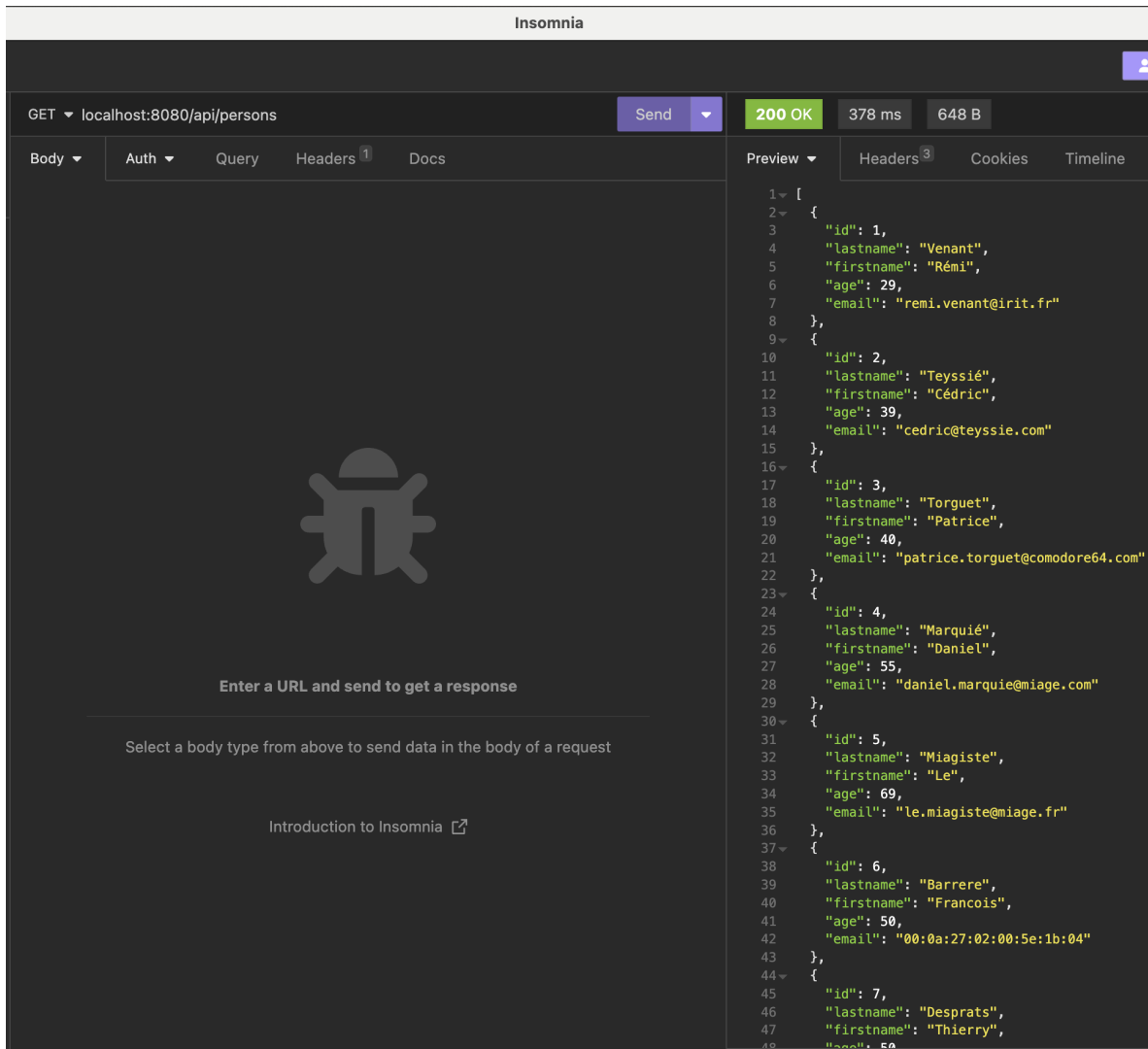


Figure 8: Test de l'app Spring Boot et de la BDD pour l'exercice 9

Sources

Dockerfile :

```
FROM maven:3.9.5-eclipse-temurin-21 AS builder
```

```
WORKDIR /app
```

```
COPY target/devops-1.0.jar ./app.jar
```

```
CMD ["java", "-jar", "app.jar"]
```

docker-compose.yml :

```
version: '3'
```

```
services:
```

```
  db:
```

```
    image: mysql:latest
```

```
    container_name: db
```

```
    volumes:
```

```
      - db_data:/var/lib/mysql
```

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

```
    environment:
```

```
      MYSQL_USER: dev
```

```
      MYSQL_DATABASE: test
```

```
      MYSQL_ROOT_PASSWORD: root
```

```
      MYSQL_PASSWORD: root
```

```
    ports:
```

```
      - "3306:3306"
```

```
    networks:
```

```
      - exo9
```

```
  app:
```

```
    build: .
```

```
    container_name: app
```

```
    ports:
```

```
      - "8080:8080"
```

```
    depends_on:
```

```
      - db
```

```
    networks:
```

```
      - exo9
```

```
  sonarqube:
```

```
    image: sonarqube:latest
```

```
    container_name: sonarqube
```

```
    ports:
```

```
      - "9000:9000"
```

```

environment:
  - SONARQUBE_JDBC_URL=jdbc:mysql://db:3306/sonarqube
    ?useUnicode=true
    &characterEncoding=utf8
    &rewriteBatchedStatements=true
    &useConfigs=maxPerformance
    &useSSL=false
  - SONARQUBE_JDBC_USERNAME=dev
  - SONARQUBE_JDBC_PASSWORD=root
networks:
  - exo9

phpmyadmin:
  image: phpmyadmin/phpmyadmin:latest
  container_name: phpmyadmin
  environment:
    - PMA_ARBITRARY=1
    - PMA_HOST=db
    - PMA_PORT=3306
  ports:
    - "8081:80"
  networks:
    - exo9

networks:
  exo9:
    driver: bridge

volumes:
  db_data:

```