AI-03139 Docker Database Consolidation

Owner	Ⓡ Rigel Arcayan
Tags	Project
Created time	@November 8, 2025 10:05 AM

I have 6 docker applications running each a db such as Mariadb and several versions of Postgres db. Is it a good practice to consolidate the databases into just one mariadb and postgres db for easy backup and recovery? How to implement if I chose to consolidate into one db each? List a step by step procedure on what change is needed in the compose file?

Yes, consolidating multiple Dockerized databases into one MariaDB and one PostgreSQL instance can simplify backup, recovery, and resource management—especially for legacy gifting and reproducibility. Here's how to implement it step by step.

🧠 Why Consolidate?

- **Simplified backups**: Fewer containers mean fewer volumes and fewer backup scripts.
- **Resource efficiency**: Reduces memory and CPU overhead from running multiple DB engines.
- **Easier recovery**: One restore point per DB engine.
- **Legacy clarity**: Easier for Brent or mentees to inherit and maintain.

However, **consolidation requires schema separation** and careful porting of data to avoid conflicts.

🛠️ Step-by-Step: Consolidating into One MariaDB and One PostgreSQL

1. **Audit Existing DB Usage**

- Identify each app's DB engine, version, and schema.
- Note DB names, users, passwords, and volume paths.

2. **Create Unique Schemas per App**

- For MariaDB: Use separate databases (`CREATE DATABASE app1_db;`)
- For PostgreSQL: Use separate schemas (`CREATE SCHEMA app1_schema;`) or databases if needed.

3. **Export Existing Data**

- Use `mysqldump` for MariaDB:

```
mysqldump -u root -p app1_db > app1_db.sql
```

- Use `pg_dump` for PostgreSQL:

```
pg_dump -U postgres -d app1_db > app1_db.sql
```

4. **Modify `docker-compose.yml`**

🔄 Replace Multiple DB Services with One Each

Before (multiple MariaDBs):

```
mariadb1:
  image: mariadb
  environment:
    MYSQL_DATABASE: app1_db
    MYSQL_ROOT_PASSWORD: rootpass
```

```
  volumes:
    - mariadb1_data:/var/lib/mysql
```

After (single MariaDB):

```
mariadb:
  image: mariadb
  environment:
    MYSQL_ROOT_PASSWORD: rootpass
  volumes:
    - mariadb_data:/var/lib/mysql
```

Before (multiple Postgres):

```
postgres1:
  image: postgres:13
  environment:
    POSTGRES_DB: app1_db
    POSTGRES_USER: user1
    POSTGRES_PASSWORD: pass1
  volumes:
    - pg1_data:/var/lib/postgresql/data
```

After (single Postgres):

```
postgres:
  image: postgres:15
  environment:
    POSTGRES_USER: masteruser
    POSTGRES_PASSWORD: masterpass
  volumes:
    - postgres_data:/var/lib/postgresql/data
```

5. **Update App Connection Strings**

- Point all apps to the consolidated DB containers.

- Use different DB names or schemas to isolate data.

Example:

```
DB_HOST=mariadb
DB_NAME=app1_db
DB_USER=root
DB_PASS=rootpass
```

6. **Import Data into Consolidated DBs**

- Use `mysql` or `psql` to import dumps into the correct schema or database.

7. **Test Each App**

- Validate DB connectivity and data integrity.
- Ensure no schema conflicts or permission issues.

8. **Backup Strategy**

- Use `mysqldump` and `pg_dumpall` to back up all schemas/databases in one go.
- Automate with cron or Docker volume snapshots.

🧭 Compose File Tips

- Use **named volumes** for persistence.
- Use **networks** to isolate DB traffic.
- Consider **labels** for legacy tracking:

  ```
  labels:
    - "codex.owner=Rigel"
    - "codex.legacy=Blindcaveman"
  ```

Would you like help drafting a zipped README for Brent that documents this consolidation protocol and includes sample compose snippets?

In each of my compose file, how do I make a connection to mariadb and postgres?

To connect your Dockerized applications to MariaDB and PostgreSQL containers defined in a `docker-compose.yml` file, you need to configure both the **database service** and the **application service** with proper environment variables and networking. Here's a structured guide:

🧩 Compose File Structure for DB Connectivity

1. **Define the Database Services**

```
services:
  mariadb:
    image: mariadb:latest
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: app_db
      MYSQL_USER: app_user
      MYSQL_PASSWORD: app_pass
    volumes:
      - mariadb_data:/var/lib/mysql
    networks:
      - backend

  postgres:
    image: postgres:latest
    environment:
      POSTGRES_DB: app_db
      POSTGRES_USER: app_user
      POSTGRES_PASSWORD: app_pass
    volumes:
      - postgres_data:/var/lib/postgresql/data
    networks:
      - backend
```

2. **Configure Application Services to Connect**

Each application must reference the correct DB host and credentials. Example for a PHP app connecting to MariaDB:

```
app1:
  image: your-app-image
  environment:
    DB_HOST: mariadb
    DB_NAME: app_db
    DB_USER: app_user
    DB_PASS: app_pass
  depends_on:
    - mariadb
  networks:
    - backend
```

Example for a Python app connecting to PostgreSQL:

```
app2:
  image: your-app-image
  environment:
    DB_HOST: postgres
    DB_NAME: app_db
    DB_USER: app_user
    DB_PASS: app_pass
  depends_on:
    - postgres
  networks:
    - backend
```

3. **Use a Shared Network**

Ensure all services are on the same Docker network:

```
networks:
  backend:
```

🧪 Connection Strings by Language

- **MariaDB (MySQL-compatible):**
  - JDBC: `jdbc:mysql://mariadb:3306/app_db`
  - PHP: `mysqli_connect('mariadb', 'app_user', 'app_pass', 'app_db')`
  - Python (SQLAlchemy): `mysql+pymysql://app_user:app_pass@mariadb/app_db`
- **PostgreSQL:**
  - JDBC: `jdbc:postgresql://postgres:5432/app_db`
  - Python (psycopg2): `postgresql://app_user:app_pass@postgres:5432/app_db`
  - Node.js (pg): `postgres://app_user:app_pass@postgres:5432/app_db`

Would you like a template compose file that consolidates multiple apps into one MariaDB and one PostgreSQL instance with schema separation?

Here is a template `docker-compose.yml` that consolidates multiple applications into one MariaDB and one PostgreSQL container, with each application using its own database or schema for isolation and maintainability.

📦 Consolidated Compose File Template

```
version: '3.8'

services:
  mariadb:
    image: mariadb:latest
    container_name: mariadb
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
    volumes:
```

```
    - mariadb_data:/var/lib/mysql
  networks:
    - backend

postgres:
  image: postgres:latest
  container_name: postgres
  environment:
    POSTGRES_USER: masteruser
    POSTGRES_PASSWORD: masterpass
  volumes:
    - postgres_data:/var/lib/postgresql/data
  networks:
    - backend

app1:
  image: your-app1-image
  environment:
    DB_HOST: mariadb
    DB_NAME: app1_db
    DB_USER: root
    DB_PASS: rootpass
  depends_on:
    - mariadb
  networks:
    - backend

app2:
  image: your-app2-image
  environment:
    DB_HOST: postgres
    DB_NAME: app2_db
    DB_USER: masteruser
    DB_PASS: masterpass
  depends_on:
    - postgres
```

```
    networks:
      - backend

  app3:
    image: your-app3-image
    environment:
      DB_HOST: postgres
      DB_NAME: app3_db
      DB_USER: masteruser
      DB_PASS: masterpass
    depends_on:
      - postgres
    networks:
      - backend

volumes:
  mariadb_data:
  postgres_data:

networks:
  backend:
```

🧭 Implementation Notes

- **MariaDB**: Create separate databases for each app using `CREATE DATABASE app1_db;`, `app2_db;`, etc.
- **PostgreSQL**: Either use separate databases (`CREATE DATABASE app2_db;`) or schemas (`CREATE SCHEMA app2_schema;`) depending on app requirements.
- **Connection strings**: Ensure each app uses the correct DB name and credentials.
- **Backup**: Use `mysqldump` and `pg_dumpall` or `pg_dump` per database for streamlined backup.

Would you like a sample SQL script to initialize the databases and users for this setup?

Here is a sample SQL initialization script to set up separate databases and users for each application in a consolidated MariaDB and PostgreSQL setup.

 MariaDB Initialization Script

```
-- Connect as root user
CREATE DATABASE app1_db;
CREATE USER 'app1_user'@'%' IDENTIFIED BY 'app1_pass';
GRANT ALL PRIVILEGES ON app1_db.* TO 'app1_user'@'%';

CREATE DATABASE app2_db;
CREATE USER 'app2_user'@'%' IDENTIFIED BY 'app2_pass';
GRANT ALL PRIVILEGES ON app2_db.* TO 'app2_user'@'%';

FLUSH PRIVILEGES;
```

 PostgreSQL Initialization Script

```
-- Connect as masteruser
CREATE DATABASE app1_db;
CREATE USER app1_user WITH ENCRYPTED PASSWORD 'app1_pass';
GRANT ALL PRIVILEGES ON DATABASE app1_db TO app1_user;

CREATE DATABASE app2_db;
CREATE USER app2_user WITH ENCRYPTED PASSWORD 'app2_pass';
GRANT ALL PRIVILEGES ON DATABASE app2_db TO app2_user;
```

Alternatively, if you prefer schema-level separation within a single PostgreSQL database:

```
-- Connect to a single shared database
CREATE SCHEMA app1_schema AUTHORIZATION app1_user;
CREATE SCHEMA app2_schema AUTHORIZATION app2_user;
```

🧩 Deployment Notes

- These scripts can be executed manually via `mysql` and `psql` CLI tools, or automated using init scripts mounted into the container's `/docker-entrypoint-initdb.d/` directory.
- Ensure your `docker-compose.yml` mounts the SQL files if using automated initialization.

Would you like a sample compose file that includes automated execution of these SQL scripts during container startup?

Here's how to automate the execution of SQL initialization scripts during container startup using Docker Compose. This setup ensures that your MariaDB and PostgreSQL containers create the necessary databases and users without manual intervention.

🪦 Directory Structure

```
project-root/
├── docker-compose.yml
├── mariadb-init/
│   └── init.sql
└── postgres-init/
    └── init.sql
```

🧾 Sample SQL Scripts

mariadb-init/init.sql

```
CREATE DATABASE app1_db;
CREATE USER 'app1_user'@'%' IDENTIFIED BY 'app1_pass';
GRANT ALL PRIVILEGES ON app1_db.* TO 'app1_user'@'%';

CREATE DATABASE app2_db;
CREATE USER 'app2_user'@'%' IDENTIFIED BY 'app2_pass';
GRANT ALL PRIVILEGES ON app2_db.* TO 'app2_user'@'%';

FLUSH PRIVILEGES;
```

postgres-init/init.sql

```
CREATE DATABASE app1_db;
CREATE USER app1_user WITH ENCRYPTED PASSWORD 'app1_pass';
GRANT ALL PRIVILEGES ON DATABASE app1_db TO app1_user;

CREATE DATABASE app2_db;
CREATE USER app2_user WITH ENCRYPTED PASSWORD 'app2_pass';
GRANT ALL PRIVILEGES ON DATABASE app2_db TO app2_user;
```

🧩 Updated `docker-compose.yml`

```
version: '3.8'

services:
  mariadb:
    image: mariadb:latest
    container_name: mariadb
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
    volumes:
      - mariadb_data:/var/lib/mysql
      - ./mariadb-init:/docker-entrypoint-initdb.d
    networks:
```

```
    - backend

  postgres:
    image: postgres:latest
    container_name: postgres
    environment:
      POSTGRES_USER: masteruser
      POSTGRES_PASSWORD: masterpass
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./postgres-init:/docker-entrypoint-initdb.d
    networks:
      - backend

  app1:
    image: your-app1-image
    environment:
      DB_HOST: mariadb
      DB_NAME: app1_db
      DB_USER: app1_user
      DB_PASS: app1_pass
    depends_on:
      - mariadb
    networks:
      - backend

  app2:
    image: your-app2-image
    environment:
      DB_HOST: postgres
      DB_NAME: app1_db
      DB_USER: app1_user
      DB_PASS: app1_pass
    depends_on:
      - postgres
    networks:
```

```
      - backend

volumes:
  mariadb_data:
  postgres_data:

networks:
  backend:
```

This setup ensures that your databases and users are created automatically when the containers are initialized for the first time. Let me know if you'd like to include schema creation or seed data in the same scripts.

Here's how to include schema creation and seed data in your automated SQL initialization scripts for MariaDB and PostgreSQL.

 Updated MariaDB Init Script (`mariadb-init/init.sql`)

```
-- Create databases and users
CREATE DATABASE app1_db;
CREATE USER 'app1_user'@'%' IDENTIFIED BY 'app1_pass';
GRANT ALL PRIVILEGES ON app1_db.* TO 'app1_user'@'%';

CREATE DATABASE app2_db;
CREATE USER 'app2_user'@'%' IDENTIFIED BY 'app2_pass';
GRANT ALL PRIVILEGES ON app2_db.* TO 'app2_user'@'%';

-- Seed data for app1_db
USE app1_db;
CREATE TABLE customers (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  email VARCHAR(100)
);
```

```
INSERT INTO customers (name, email) VALUES ('Alice', 'alice@example.co
m'), ('Bob', 'bob@example.com');

-- Seed data for app2_db
USE app2_db;
CREATE TABLE products (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  price DECIMAL(10,2)
);
INSERT INTO products (name, price) VALUES ('Widget', 19.99), ('Gadget', 29.9
9);

FLUSH PRIVILEGES;
```

 Updated PostgreSQL Init Script (`postgres-init/init.sql`)

```
-- Create databases and users
CREATE DATABASE app1_db;
CREATE USER app1_user WITH ENCRYPTED PASSWORD 'app1_pass';
GRANT ALL PRIVILEGES ON DATABASE app1_db TO app1_user;

CREATE DATABASE app2_db;
CREATE USER app2_user WITH ENCRYPTED PASSWORD 'app2_pass';
GRANT ALL PRIVILEGES ON DATABASE app2_db TO app2_user;

-- Seed data for app1_db
\connect app1_db
CREATE TABLE employees (
  id SERIAL PRIMARY KEY,
  name VARCHAR(100),
  department VARCHAR(100)
);
INSERT INTO employees (name, department) VALUES ('John Doe', 'HR'), ('Jan
```

```
e Smith', 'Engineering');

-- Seed data for app2_db
\connect app2_db
CREATE TABLE orders (
  id SERIAL PRIMARY KEY,
  product_name VARCHAR(100),
  quantity INT
);
INSERT INTO orders (product_name, quantity) VALUES ('Widget', 3), ('Gadge
t', 5);
```

These scripts will automatically run when the containers are initialized for the first time, creating databases, users, tables, and inserting sample data. Let me know if you'd like to add constraints, indexes, or foreign keys to these schemas.