Documentation: Deployment of Moodle and MariaDB Using Docker Compose

1 Deployment Explanation and Data Management

This document explains the deployment and management strategy used in the `docker-compose.yml` file for the Moodle Learning Management System (LMS) and MariaDB database services.

1.1 Services Overview

1.1.1 MariaDB Service

- **Image**: `bitnami/mariadb:11.4`
  The MariaDB service uses the Bitnami MariaDB image, which provides a reliable and production-ready database engine.
- **Environment Variables**:
  - `ALLOW_EMPTY_PASSWORD=yes`: Allows starting the service without a password (intended for development only).
  - `MARIADB_USER=bn_moodle`: Defines the database user for Moodle.
  - `MARIADB_DATABASE=bitnami_moodle`: Specifies the Moodle database name.
  - `MARIADB_CHARACTER_SET=utf8mb4` and `MARIADB_COLLATE=utf8mb4_unicode_ci`: Ensure proper character encoding for multilingual support.
- **Volume Configuration**:
  - `mariadb_data:/bitnami/mariadb`: Stores persistent database files, ensuring that data remains intact across container restarts or updates.

1.1.2 Moodle Service

- **Image**: `bitnami/moodle:4.5`
  The Moodle service uses Bitnami's Moodle image, providing a preconfigured environment for deploying Moodle quickly.

- **Ports**:
  - `80:8080`: Maps the container's HTTP service to port 80 on the host.
  - `443:8443`: Maps the container's HTTPS service to port 443 on the host.
- **Environment Variables**:
  - `MOODLE_DATABASE_HOST=mariadb`: Specifies the hostname of the MariaDB service.
  - `MOODLE_DATABASE_PORT_NUMBER=3306`: Indicates the port for the MariaDB connection.
  - `MOODLE_DATABASE_USER=bn_moodle`: Moodle will use this user to connect to the database.
  - `MOODLE_DATABASE_NAME=bitnami_moodle`: Sets the database name for Moodle.
  - `ALLOW_EMPTY_PASSWORD=yes`: Allows Moodle to connect to MariaDB without a password (development only).
- **Volume Configuration**:
  - `moodle_data:/bitnami/moodle`: Stores Moodle core files.
  - `moodledata_data:/bitnami/moodledata`: Stores Moodle user data (uploads, cache, etc.).
- **Dependency**:
  - `depends_on: mariadb`: Ensures that the MariaDB service starts before Moodle to avoid connection errors during initialization.

1.2 Data Management Strategy

1.2.1 Volumes for Data Persistence

Three volumes are defined to ensure data persistence:

- `mariadb_data`: Stores all database-related files to retain data across restarts or updates.
- `moodle_data`: Holds the Moodle application files for maintaining consistency in custom configurations or plugins.
- `moodledata_data`: Contains Moodle's uploaded files and cache to avoid data loss between container lifecycles.

1.2.2 Backup Strategy

- **Database Backups**: Regular backups can be created using MariaDB's native `mysqldump` tool, run either manually or via cron jobs in a separate backup container.
- **File Backups**: Moodle data directories (`moodle_data` and `moodledata_data`) can be backed up using file synchronization tools like `rsync` or through automated cloud backup solutions.

1.3 Deployment Instructions

1. **Run the services**: Start the services using:

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

2. **Check the status**: Verify that both MariaDB and Moodle services are running:

   ```
   docker-compose ps
   ```

3. **Access Moodle**: Open a web browser and navigate to `http://localhost` to access the Moodle platform.

1.4 Additional Recommendations

- **Security**: In production, replace `ALLOW_EMPTY_PASSWORD` with secure database credentials managed through environment variables or Docker secrets.
- **Scaling**: Use Docker Swarm or Kubernetes for high availability by scaling the Moodle service horizontally.

This deployment ensures a straightforward setup while maintaining flexibility for future enhancements.