**Dockerizing a Django Application with MariaDB**

Dockerizing a Django application with MariaDB involves packaging your Django web application and its associated MariaDB database into Docker Containers.

This document will guide you through the process of creating Docker containers for Django app and MariaDB, along with explanations of the steps involved.

**Table of Contents:**

1. Prerequisites

- Docker Installation

- Django Application

- Database Setup

2. Dockerizing the Django Application

- Creating a Dockerfile

- Build a Custom Django Image

- Docker Compose Configuration

3. Running the Docker Containers

- Docker Compose Up

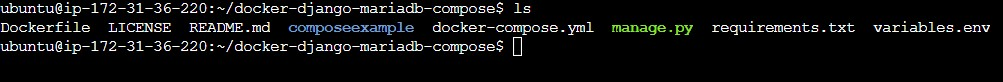
- Accessing the Django Application

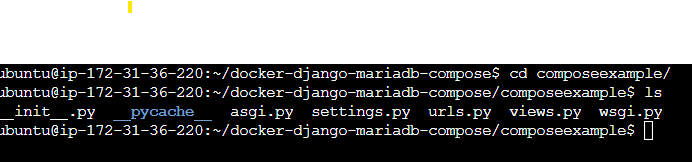
**1)Django Application:**

**Source:** [**https://github.com/ramyacloud001/docker-django-mariadb-compose**](https://github.com/ramyacloud001/docker-django-mariadb-compose)

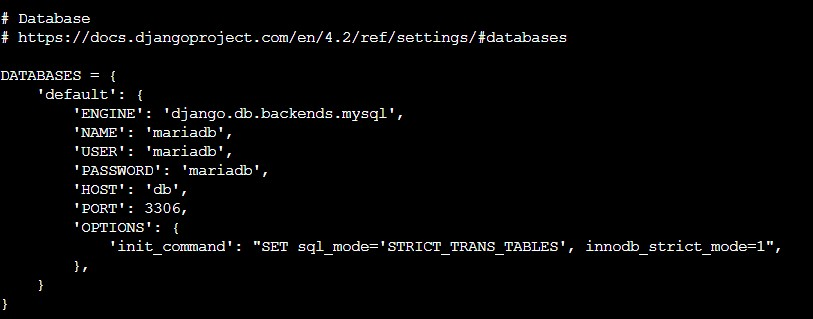
* $ git clone https://github.com/damufo/docker-django-mariadb-compose.git
* $ cd docker-django-mariadb-compose
* $ docker-compose run web django-admin startproject composeexample .
* $ sudo chown -R $USER:$USER .

Compose example folder will be created with list of files as shown in the below images,





Edit file composeexample/settings.py and set database section as:



**Update ALLOWED\_HOSTS:**

Open your Django project's settings file (settings.py) and locate the ALLOWED\_HOSTS setting. Update it to include the hostname or IP address that corresponds to the HTTP\_HOST header in your request.



**2)Dockerizing the Django Application**

**2.1)Dockerfile :**

To containerize your Django application, we need to create a `Dockerfile` in your project's root directory. A `Dockerfile` is used to define how your Docker image should be built.

* PYTHONUNBUFFERED environment variable to ensure that Python's standard output is not buffered. This is useful for logging and debugging in a containerized environment.
* Created a directory/requirement in the container to store Python package requirements.
* The requirements.txt file is copied into the container, and pip is used to install the required Python packages.
* created a directory /code in the container to hold your Django project code.
* Django application code is copied to the /code directory inside the container
* Finally, defined the command that will run your Django application using the development server (runserver) on port 8000.

**2.2)Build a Custom Django Image:** Building a custom Docker image for your Django application.

**2.3)Docker Compose Configuration:**

Created a `docker-compose.yml` file to define the services for your Django app and the MariaDB database.

Docker-compose file consists of

**db service:**

* Uses the official MariaDB Docker image.
* Loads environment variables from a variables.env file.
* Sets up a volume named mariadb\_data to persist MariaDB data.

**adminer service:**

* Uses the official Adminer Docker image.
* Always restarts the container.
* Maps port 8080 of the host to port 8080 of the container, allowing you to access the Adminer web interface.

**web service:**

* Builds the Docker image using the local context (the current directory).
* Defines the command to run the Django development server.
* Mounts the current directory (.) into the /code directory inside the container, allowing live code reloading.
* Maps port 8009 of the host to port 8000 of the container for accessing the Django application.
* Depends on the db service, ensuring that the database service is started before the Django application.

**mariadb\_data:**

* Defines the volume mariadb\_data for persisting MariaDB data.

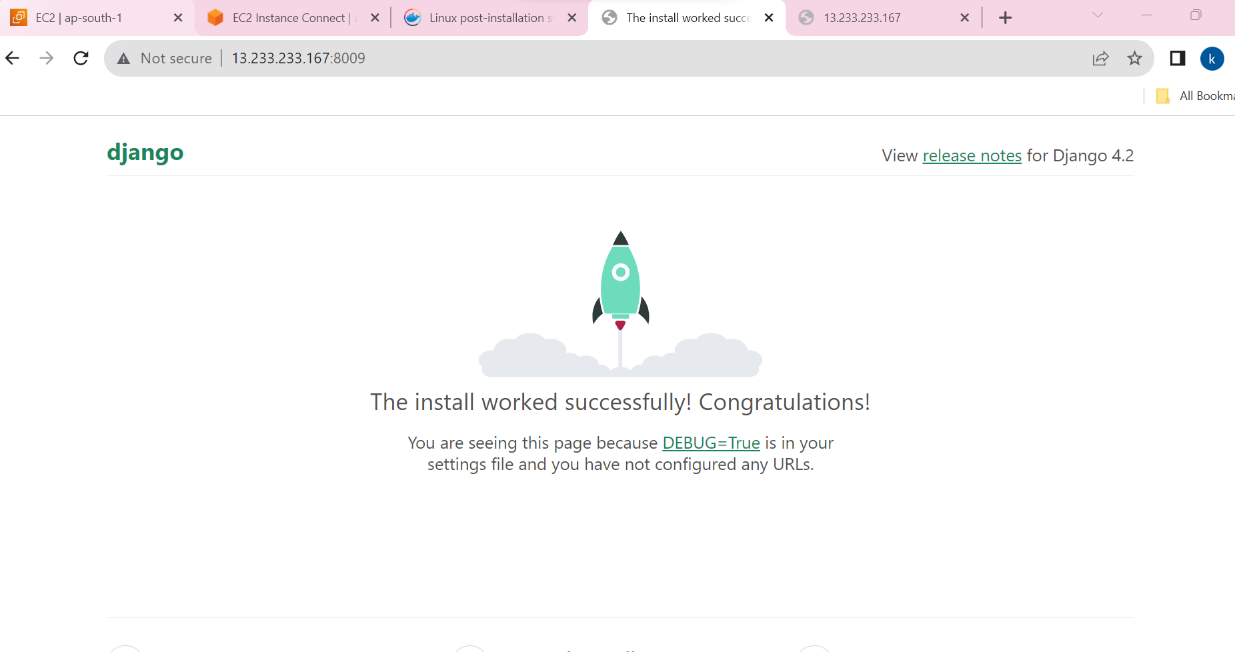
Run the following command in the directory containing the docker-compose.yml:

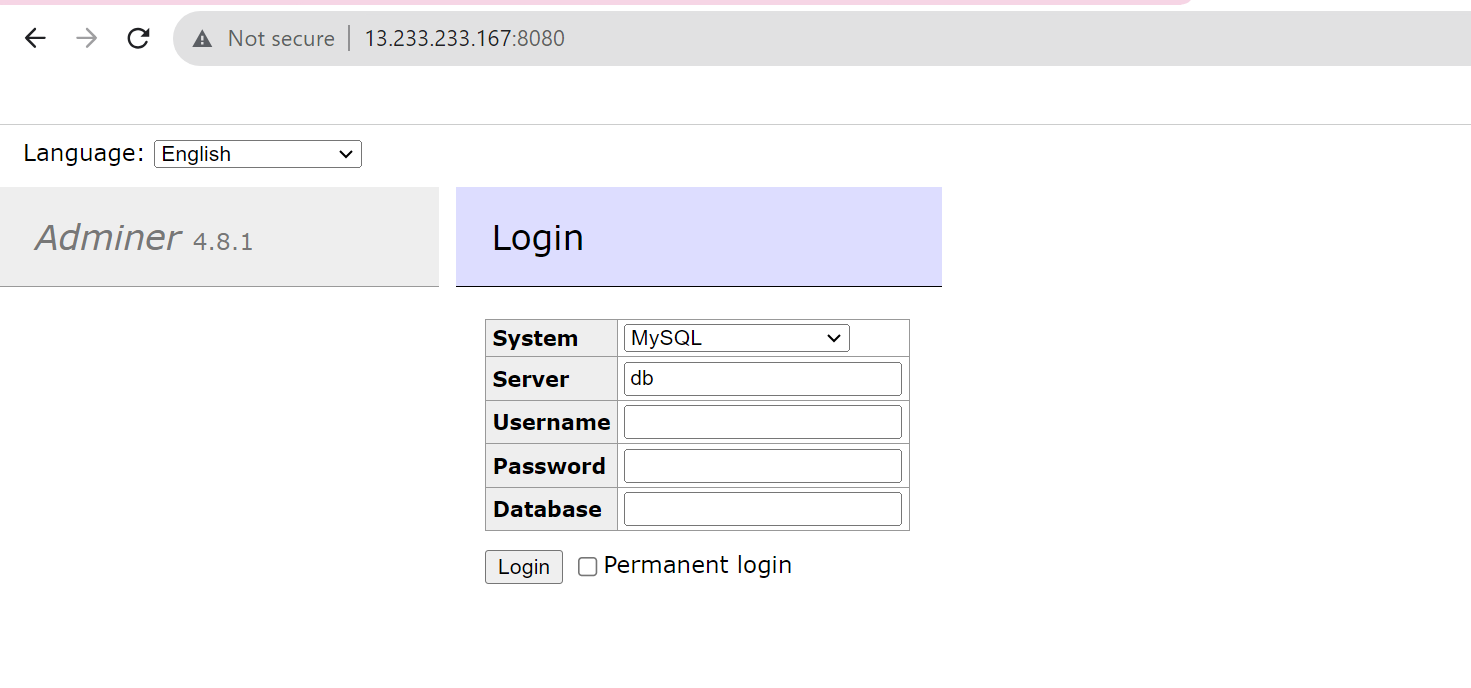
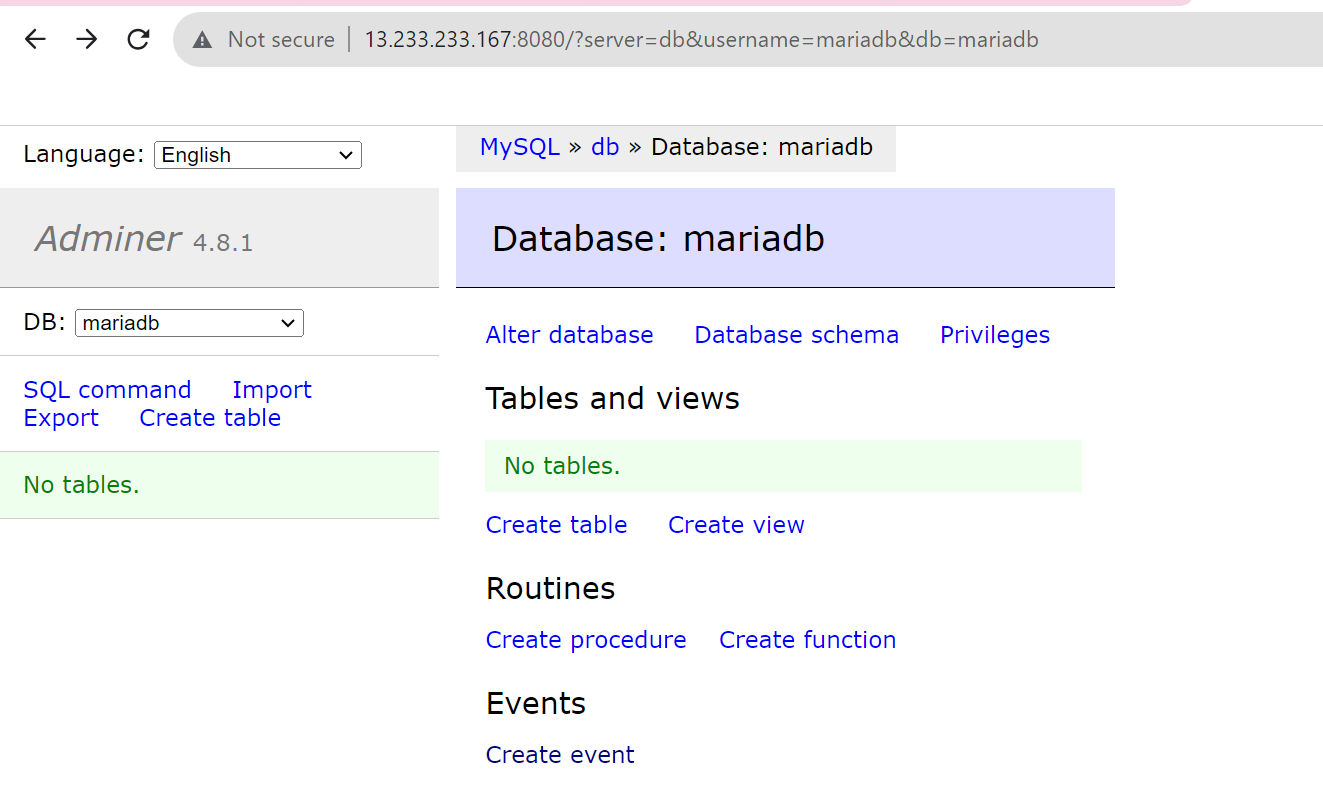
**docker-compose up**

**3)Accessing the Django Application:**

**3.1)You can access your Django application by navigating to** [**http://localhost:8009 (in**](http://localhost:8009%20(in) **my case I used aws ec2 instance ipaddress) in your web browser.**

**3.2)Adminer at http://localhost:8080 to manage the MariaDB database.**





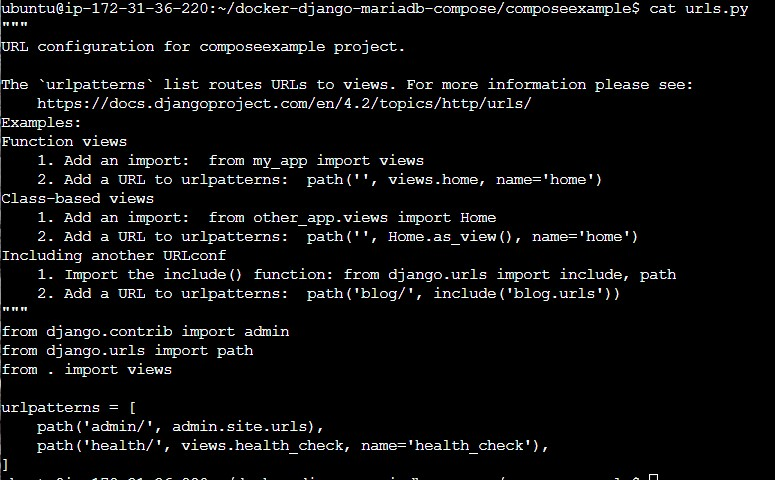
**4)Set up a health check to ensure the Django application is responding correctly.**

For a Django application, a health check is typically implemented as an HTTP endpoint that returns a response indicating the health status of the application. This status endpoint can be used for monitoring and to ensure that the application is functioning correctly.

4.1)Created views.py file

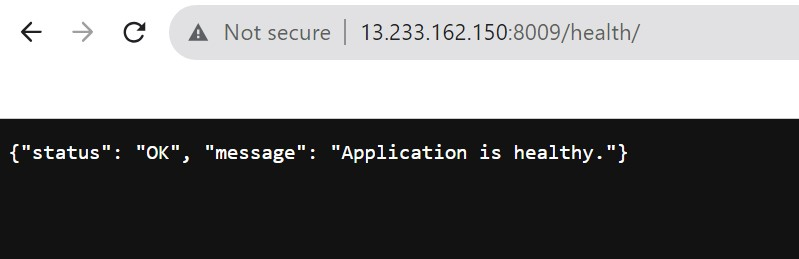


4.2) Made changes in url.py file



Test the Health Check:test the health check endpoint by making an HTTP request to it.

* If the application is healthy, you should receive an HTTP 200 OK response with the JSON response containing {'status': 'OK'.



**5. Create a Docker Volume for MariaDB:**

To create a Docker volume named "mariadb\_data" for persisting MariaDB data across container restarts, you can leave your existing volumes section as it is in your docker-compose.yml:

volumes:

mariadb\_data:

Docker Compose will automatically create the "mariadb\_data" volume when you run docker-compose up, and it will persist the MariaDB data in this volume.