Deployment Guide - Actual Commands

# 1. How to Build and Run the Project Locally Using Docker Compose

### Prerequisites:

Ensure Docker and Docker Compose are installed on your system.

### Steps to Run the Project Locally

1. Clone the repository (if not done already):

```bash  
git clone <repository\_url>  
cd <project\_directory>  
```

2. Navigate to the project folder where the docker-compose.yml file is located:

```bash  
cd /path/to/your/project/directory  
```

3. Build and start the services using Docker Compose:

```bash  
docker-compose up --build -d  
```

4. Verify that the containers are running:

```bash  
docker ps  
```

5. Test the application locally by using curl:

```bash  
curl -X POST -H "Content-Type: application/json" -d '{"username": "testuser", "password": "password123"}' http://localhost:5000/login  
```

# 2. How to Deploy the Project to AWS

### Prerequisites:

An AWS account and EC2 instance with appropriate security group settings to allow traffic on port 5000 (or your desired port).

### Steps to Deploy on AWS

1. Launch an EC2 instance (Ubuntu or Amazon Linux).

2. SSH into the EC2 Instance:

```bash  
ssh -i "C:\Users\Anurag\Desktop\lembda.pem" ec2-user@ec2-52-66-240-107.ap-south-1.compute.amazonaws.com  
```

3. Install Docker and Docker Compose on EC2:

```bash  
sudo yum install docker -y # For Amazon Linux  
sudo systemctl start docker  
sudo systemctl enable docker  
  
# Install Docker Compose  
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose  
sudo chmod +x /usr/local/bin/docker-compose  
```

4. Transfer the Project Files to EC2:

```bash  
scp -i "C:\Users\Anurag\Desktop\lembda.pem" -r /path/to/your/project/directory ec2-user@ec2-52-66-240-107.ap-south-1.compute.amazonaws.com:/home/ec2-user  
```

5. Navigate to the Project Directory on EC2:

```bash  
cd /home/ec2-user/Blog\ Platform  
```

6. Build and Start the Docker Containers:

```bash  
docker-compose up --build -d  
```

7. Verify the Application is Running:

```bash  
docker ps  
```

8. Access the Application:

Open your browser and visit the public IP of your EC2 instance on the appropriate port:  
```bash  
http://52.66.240.107:5000  
```

# 3. Live Deployment

### Public URL for Deployed Application

Once deployed, the application should be accessible via:  
```bash  
http://52.66.240.107:5000  
```

To secure the URL with HTTPS:

1. Install Certbot:

```bash  
sudo yum install certbot  
```

2. Obtain SSL Certificate with Certbot:

```bash  
sudo certbot certonly --standalone -d <your\_domain>  
```

3. Configure Nginx for HTTPS (optional for reverse proxy setup).

# 4. API Documentation

## 4.1 Register Endpoint

\*\*URL\*\*: /register/  
\*\*Method\*\*: POST  
\*\*Body\*\*:  
```json  
{  
"username": "testuser",  
"password": "password123"  
}  
```

\*\*Response\*\*:  
- \*\*Success\*\*: 200 OK  
 ```json  
 { "message": "User registered successfully" }  
 ```  
- \*\*Failure\*\*: 400 Bad Request  
 ```json  
 { "message": "Username and password are required" }  
 ```

## 4.2 Login Endpoint

\*\*URL\*\*: /login/  
\*\*Method\*\*: POST  
\*\*Body\*\*:  
```json  
{  
"username": "testuser",  
"password": "password123"  
}  
```

\*\*Response\*\*:  
- \*\*Success\*\*: 200 OK  
 ```json  
 { "message": "Login successful" }  
 ```  
- \*\*Failure\*\*: 401 Unauthorized  
 ```json  
 { "message": "Invalid credentials" }  
 ```

# 5. Trade-offs and Design Decisions

### Database Choice

SQLite is used for simplicity in local development.  
- \*\*Trade-off\*\*: Not suitable for large-scale applications. In production, PostgreSQL or MySQL would be better.

### Security

Passwords are hashed using `werkzeug.security.generate\_password\_hash`.  
- \*\*Trade-off\*\*: No advanced session management (e.g., JWT or OAuth2). Consider adding token-based authentication for production.

### Docker Compose for Local Development

Docker Compose is used for containerizing the application, simplifying local development and ensuring all services run together.