Deployment Guide: Course Allocation Backend

1. Introduction

This document provides a comprehensive, step-by-step guide for deploying the Course Allocation Backend application on a Linux server using a MySQL database. This guide is written for a server administrator and assumes minimal prior knowledge of the application stack.

The production environment will consist of:

- **Nginx:** As a reverse proxy to handle incoming web traffic.
- **Gunicorn:** As a WSGI server to run the Python Flask application.
- **MySQL:** As the production database.
- **Systemd:** To manage the application process and ensure it runs continuously.

This guide primarily uses commands for **Debian/Ubuntu**. Notes are provided for **Fedora/RHEL** where commands differ.

---

2. Prerequisites

2.1. Server Preparation

1. **Access Your Server:** Connect to your Linux server via SSH.
2. **Update System Packages:** Ensure your server's package list and installed packages are up-to-date.

   ```
   # For Debian/Ubuntu
   sudo apt update && sudo apt upgrade -y

   # For Fedora/RHEL
   sudo dnf update -y
   ```

3. **Install Essential Tools:** Install Python, the virtual environment tool, development tools for building packages, and Git.

   ```
   # For Debian/Ubuntu
   sudo apt install -y python3 python3-venv python3-dev build-essential git

   # For Fedora/RHEL
   sudo dnf install -y python3 python3-devel @development-tools git
   ```

2.2. Create a Dedicated Application User

For security, the application should not run as the `root` user. Let's create a dedicated user account named `course_app`.

```
sudo adduser course_app
```

You will be prompted to set a password and fill in user information. You can leave the information fields blank by pressing `Enter`.

---

3. Database Setup (MySQL)

We will use MySQL as the production database.

1. **Install MySQL Server:**

   ```
   # For Debian/Ubuntu
   sudo apt install -y mysql-server

   # For Fedora/RHEL
   sudo dnf install -y mysql-server
   sudo systemctl start mysqld
   sudo systemctl enable mysqld
   ```

2. **Secure the MySQL Installation:** Run the security script included with MySQL. It will ask you to set a root password, remove anonymous users, and disable remote root login. It is highly recommended to answer 'Yes' to these prompts.

   ```
   sudo mysql_secure_installation
   ```

3. **Create the Database and Database User:** Log in to the MySQL interactive terminal as the root user.

   ```
   sudo mysql
   ```

Now, run the following SQL commands to create the database and a user for our application. **Replace `'a_very_strong_password'` with a secure password of your choice.**

```
-- Create the database for the application
CREATE DATABASE course_allocation_db;

-- Create a dedicated user for the application that can only connect from localhost
CREATE USER 'course_app_user'@'localhost' IDENTIFIED BY 'a_very_strong_password';

-- Grant all privileges on the new database to the new user
GRANT ALL PRIVILEGES ON course_allocation_db.* TO 'course_app_user'@'localhost';

-- Apply the changes
FLUSH PRIVILEGES;

-- Exit the mysql terminal
EXIT;
```

---

4. Application Setup

Now we will switch to our dedicated user to set up the application code.

1. **Log in as the Application User:**

   ```
   su - course_app
   ```

2. **Clone the Application Repository:** Clone the project code from its Git repository into the user's home directory.

   ```
   # Replace <repository_url> with the actual Git URL
   git clone https://github.com/Ekediee/course-allocation-backend.git /home/course_app/course-allocation-backend
   ```

   Note: If you are moving the files manually, ensure they are placed in `/home/course_app/course-allocation-backend`.

3. **Navigate to the Project Directory:**

   ```
   cd /home/course_app/course-allocation-backend
   ```

4. **Create and Activate a Python Virtual Environment:** A virtual environment keeps the project's dependencies isolated.

   ```
   python3 -m venv venv
   source venv/bin/activate
   ```

   Your shell prompt should now be prefixed with `(venv)`.

5. **Install Dependencies:** Install the required Python packages from `requirements.txt` and add Gunicorn for serving the application.

   ```
   pip install -r requirements.txt gunicorn
   ```

---

5. Application Configuration

The application uses a `.env` file for configuration. We need to create one for the production environment.

1. **Create the `.env` file:**

   ```
   nano .env
   ```

2. **Add Production Configuration:** Copy and paste the following content into the file. **You must replace the placeholder values.**

   ```
   # Flask Configuration
   FLASK_APP=run.py
   FLASK_ENV=production

   # IMPORTANT: Generate a strong, random key for security
   # Use the command: openssl rand -hex 32
   SECRET_KEY=your_generated_secret_key

   # Database Connection URL for MySQL
   # Use the password you created in Step 3.3
   DATABASE_URL=mysql+pymysql://course_app_user:a_very_strong_password@localhost/course_allocation_db
   ```

```
# JWT (Authentication) Configuration
# IMPORTANT: Generate another strong, random key
JWT_SECRET_KEY=your_generated_jwt_secret_key
JWT_COOKIE_SECURE=True
JWT_COOKIE_CSRF_PROTECT=True
```

- To generate the required secret keys, you can use `openssl rand -hex 32` in your terminal.
- Press `Ctrl+X`, then `Y`, then `Enter` to save and exit nano.

5.1. Email Configuration (Important)

For features that require sending emails (such as password resets or notifications), you must configure the email server settings in the `.env` file. If email is not configured, these features will not work.

Add the following variables to your `.env` file, customized for your email provider (e.g., Gmail, SendGrid, or your own SMTP server):

```
# Email Sending Configuration
MAIL_SERVER=smtp.example.com
MAIL_PORT=587
MAIL_USE_TLS=True
MAIL_USE_SSL=False
MAIL_USERNAME=your-email@example.com
MAIL_PASSWORD=your-email-password-or-app-key
MAIL_DEFAULT_SENDER=("Your App Name" <noreply@example.com>)
```

Notes:

- `MAIL_USE_TLS` and `MAIL_USE_SSL` are mutually exclusive. Use the one required by your provider.
- For services like Gmail, you may need to generate an "App Password" instead of using your regular account password.
- The `MAIL_DEFAULT_SENDER` is the "From" address that will appear on outgoing emails.

6. Run Database Migrations

With the configuration in place, we can now set up the database schema using the existing migrations.

```
# Ensure your virtual environment is still active
flask db upgrade
```

This command will connect to the MySQL database and create all the necessary tables.

---

7. Setting Up the Gunicorn Service

To ensure the application runs reliably in the background, we will create a `systemd` service file for Gunicorn.

1. **Exit from the `course_app` user session:**

   ```
   exit
   ```

   You should now be back in your `sudo` user session.

2. **Create the Systemd Service File:**

   ```
   sudo nano /etc/systemd/system/course-allocation.service
   ```

3. **Add the Service Configuration:** Copy and paste the following content into the file.

   ```
   [Unit]
   Description=Gunicorn instance to serve the Course Allocation Backend
   After=network.target

   [Service]
   User=course_app
   Group=www-data
   WorkingDirectory=/home/course_app/course-allocation-backend
   Environment="PATH=/home/course_app/course-allocation-backend/venv/bin"
   ExecStart=/home/course_app/course-allocation-backend/venv/bin/gunicorn --workers 3 --bind unix:course-allocation.sock -m (

   [Install]
   WantedBy=multi-user.target
   ```

4. **Start and Enable the Service:**

```
sudo systemctl start course-allocation
sudo systemctl enable course-allocation
```

5. **Check the Service Status:** Verify that the service started without errors.

   ```
   sudo systemctl status course-allocation
   ```

   Look for a green `active (running)` status. If there are errors, you can check the logs with `sudo journalctl -u course-allocation`. Press `Q` to exit the status/log view.

---

8. Configure Nginx as a Reverse Proxy

Nginx will sit in front of our application, handling web traffic and forwarding it to Gunicorn.

1. **Install Nginx:**

   ```
   # For Debian/Ubuntu
   sudo apt install -y nginx

   # For Fedora/RHEL
   sudo dnf install -y nginx
   ```

2. **Create an Nginx Configuration File:**

   ```
   sudo nano /etc/nginx/sites-available/course-allocation
   ```

3. **Add the Nginx Configuration:** Copy and paste the following, replacing `your_domain_or_server_ip` with your server's public domain name or IP address.

   ```
   server {
       listen 80;
       server_name your_domain_or_server_ip;

       location / {
           include proxy_params;
           proxy_pass http://unix:/home/course_app/course-allocation-backend/course-allocation.sock;
       }
   }
   ```

4. **Enable the Nginx Site:** Link the configuration file to the `sites-enabled` directory.

   ```
   sudo ln -s /etc/nginx/sites-available/course-allocation /etc/nginx/sites-enabled
   # It's good practice to remove the default config
   sudo rm /etc/nginx/sites-enabled/default
   ```

5. **Test and Restart Nginx:**

   ```
   sudo nginx -t  # Test the configuration for syntax errors
   sudo systemctl restart nginx
   ```

6. **Adjust Firewall (if applicable):** If you are using a firewall like `ufw`, allow traffic to Nginx.

   ```
   sudo ufw allow 'Nginx Full'
   ```

---

9. Final Verification

The deployment should now be complete.

- **Access the Application:** Open a web browser and navigate to `http://your_domain_or_server_ip`. You should not see a "Welcome to Nginx" page or a 502 Bad Gateway error. While the backend may not have a frontend to display, a successful connection means the services are working together.
- **Troubleshooting:**
  - **502 Bad Gateway Error:** This usually means Nginx can't communicate with Gunicorn. Check the status of the `course-allocation` service (`sudo systemctl status course-allocation`) and its logs (`sudo journalctl -u course-allocation`).
  - **Other Errors:** Check the Nginx error logs at `/var/log/nginx/error.log`.

Your application is now deployed and running.

---

10. Future Updates and Scaling

As the application evolves, you will need to deploy updates. This section covers the process for deploying new code and scaling the application to handle more traffic.

10.1. Deploying Application Updates

Follow these steps to update the application with new code from the repository.

1. **Log in as the Application User:**

   ```
   su - course_app
   ```

2. **Navigate to the Project Directory:**

   ```
   cd /home/course_app/course-allocation-backend
   ```

3. **Pull the Latest Code:** Fetch the latest changes from the main branch of your repository.

   ```
   git pull origin main
   ```

   (Note: Replace `main` with your primary branch name if it's different, e.g., `master`)

4. **Activate the Virtual Environment:**

   ```
   source venv/bin/activate
   ```

5. **Update Dependencies:** If new packages have been added to `requirements.txt`, install them.

   ```
   pip install -r requirements.txt
   ```

6. **Run New Database Migrations:** If the update includes changes to the database model, apply the new migrations.

   ```
   flask db upgrade
   ```

7. **Exit and Restart the Service:** Log out of the `course_app` user session and restart the application service to apply all changes.

   ```
   exit
   sudo systemctl restart course-allocation
   ```

10.2. Scaling the Application

There are two primary ways to scale the application:

A. Vertical Scaling

This involves increasing the resources (CPU, RAM) of the single server where the application is hosted. This is the simplest approach but has limits. If you find that the server's CPU or memory usage is consistently high, you may need to upgrade your server plan.

B. Horizontal Scaling

This involves distributing the load across multiple instances of the application.

1. **Increase Gunicorn Workers:** The simplest form of horizontal scaling is to increase the number of Gunicorn worker processes on your existing server. This allows the application to handle more concurrent requests.

   - **Edit the systemd service file:**

     ```
     sudo nano /etc/systemd/system/course-allocation.service
     ```

   - **Adjust the `--workers` flag:** A common recommendation is `(2 * number_of_cpu_cores) + 1`. For a 2-core server, you might use 5 workers.

     ```
     # Example for 5 workers
     ExecStart=/home/course_app/course-allocation-backend/venv/bin/gunicorn --workers 5 --bind unix:course-allocation.soc
     ```

   - **Reload and restart the service:**

     ```
     sudo systemctl daemon-reload
     sudo systemctl restart course-allocation
     ```

2. **Deploying to Multiple Servers:** For very high traffic, you can deploy the application across multiple identical servers. This involves:

   - Provisioning new servers and following this entire deployment guide on each one.
   - Ensuring all servers connect to the **same central MySQL database**. The database should be hosted on a dedicated, powerful server that all application servers can access.

- Setting up a **Load Balancer** (e.g., using Nginx, HAProxy, or a cloud provider's load balancer) to distribute incoming traffic evenly across your application servers.