Travel Booking Application - Deployment Guide

Deployment Options

This guide provides step-by-step instructions for deploying the Travel Booking Application to various cloud platforms.

---

1. PythonAnywhere Deployment

Prerequisites

- PythonAnywhere account (free tier available)
- Basic knowledge of web hosting

Step-by-Step Instructions

Step 1: Upload Project Files

```
# On your local machine, create a zip of the project
zip -r travel-booking-app.zip travel-booking-app/

# Upload the zip file to PythonAnywhere via Files tab
```

Step 2: Extract and Setup

```
# In PythonAnywhere bash console
cd /home/yourusername/
unzip travel-booking-app.zip
cd travel-booking-app/
```

Step 3: Create Virtual Environment

```
# Create virtual environment
python3.11 -m venv venv
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt
```

Step 4: Setup Database

```
# Create .env file
cp .env.sample .env
# Edit .env with your settings

# Setup database
python manage.py setup_database
```

Step 5: Configure WSGI Create `/var/www/yourusername_pythonanywhere_com_wsgi.py`:

```
import os
import sys

# Add your project directory to Python path
path = '/home/yourusername/travel-booking-app'
if path not in sys.path:
    sys.path.append(path)

# Set Django settings module
os.environ['DJANGO_SETTINGS_MODULE'] = 'travel_booking.settings'

# Import Django WSGI application
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
```

Step 6: Configure Web App

1. Go to Web tab in PythonAnywhere dashboard
2. Create new web app
3. Choose Manual configuration
4. Set Python version to 3.11
5. Set source code directory: `/home/yourusername/travel-booking-app`
6. Set working directory: `/home/yourusername/travel-booking-app`
7. Edit WSGI configuration file with above content

Step 7: Configure Static Files In Web tab, add static files mapping: - URL: `/static/` - Directory: `/home/yourusername/travel-booking-app/static/`

Step 8: Environment Variables In Web tab, add environment variables: - `SECRET_KEY`: Your secret key - `DEBUG`: False - `DATABASE_ENGINE`: django.db.backends.mysql (if using MySQL)

Step 9: Database Setup (MySQL)

```
# In PythonAnywhere MySQL console
CREATE DATABASE yourusername$travel_booking;
```

Update `.env`:

```
DATABASE_ENGINE=django.db.backends.mysql
DATABASE_NAME=yourusername$travel_booking
DATABASE_USER=yourusername
DATABASE_PASSWORD=your_mysql_password
DATABASE_HOST=yourusername.mysql.pythonanywhere-services.com
```

Step 10: Final Steps

```
# Collect static files
python manage.py collectstatic --noinput

# Run migrations
python manage.py migrate

# Create superuser
python manage.py createsuperuser
```

---

2. AWS Deployment (EC2 + RDS)

Prerequisites

- AWS account
- Basic knowledge of AWS services
- SSH client

Architecture

- **EC2**: Web server hosting
- **RDS**: MySQL database
- **S3**: Static files storage
- **CloudFront**: CDN for static files

Step-by-Step Instructions

Step 1: Setup RDS Database

```
# Create RDS MySQL instance
# Note down endpoint, username, password
```

Step 2: Launch EC2 Instance

```
# Launch Ubuntu 22.04 LTS instance
# Configure security group (ports 80, 443, 22)
# Connect via SSH
```

Step 3: Server Setup

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install Python, pip, nginx
sudo apt install python3 python3-pip python3-venv nginx git mysql-client -y
```

```
# Install MySQL client libraries
sudo apt install python3-dev default-libmysqlclient-dev build-essential -y
```

Step 4: Application Setup

```
# Clone repository
git clone <your-repo-url> /home/ubuntu/travel-booking-app
cd /home/ubuntu/travel-booking-app

# Create virtual environment
python3 -m venv venv
source venv/bin/activate

# Install dependencies
pip install -r requirements.txt
pip install gunicorn
```

Step 5: Environment Configuration

```
# Create .env file
cp .env.sample .env

# Edit .env with RDS details
DATABASE_ENGINE=django.db.backends.mysql
DATABASE_NAME=travel_booking
DATABASE_USER=admin
DATABASE_PASSWORD=your_rds_password
DATABASE_HOST=your-rds-endpoint.amazonaws.com
DATABASE_PORT=3306

DEBUG=False
ALLOWED_HOSTS=your-domain.com,your-ec2-ip
```

Step 6: Database Setup

```
# Setup database
python manage.py setup_database
python manage.py collectstatic --noinput
```

Step 7: Gunicorn Service Create `/etc/systemd/system/travel-booking.service`:

```
[Unit]
Description=Travel Booking Django App
After=network.target

[Service]
User=ubuntu
```

```
Group=ubuntu
WorkingDirectory=/home/ubuntu/travel-booking-app
Environment="PATH=/home/ubuntu/travel-booking-app/venv/bin"
ExecStart=/home/ubuntu/travel-booking-app/venv/bin/gunicorn --workers 3 --bind unix:/home/ub

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

Step 8: Nginx Configuration Create `/etc/nginx/sites-available/travel-booking`:

```
server {
    listen 80;
    server_name your-domain.com your-ec2-ip;

    location /static/ {
        alias /home/ubuntu/travel-booking-app/static/;
    }

    location / {
        proxy_pass http://unix:/home/ubuntu/travel-booking-app/travel_booking.sock;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Step 9: Start Services

```
# Enable and start services
sudo systemctl daemon-reload
sudo systemctl enable travel-booking
sudo systemctl start travel-booking

# Enable nginx
sudo ln -s /etc/nginx/sites-available/travel-booking /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

---

3. Docker Deployment

Dockerfile

```
FROM python:3.11-slim
```

```
WORKDIR /app

COPY requirements.txt .
RUN pip install -r requirements.txt

COPY . .

EXPOSE 8000

CMD ["gunicorn", "--bind", "0.0.0.0:8000", "travel_booking.wsgi:application"]
```

Docker Compose

```
version: '3.8'

services:
  web:
    build: .
    ports:
      - "8000:8000"
    environment:
      - DATABASE_ENGINE=django.db.backends.mysql
      - DATABASE_NAME=travel_booking
      - DATABASE_USER=travel_user
      - DATABASE_PASSWORD=password
      - DATABASE_HOST=db
    depends_on:
      - db

  db:
    image: mysql:8.0
    environment:
      - MYSQL_DATABASE=travel_booking
      - MYSQL_USER=travel_user
      - MYSQL_PASSWORD=password
      - MYSQL_ROOT_PASSWORD=rootpassword
    ports:
      - "3306:3306"
    volumes:
      - mysql_data:/var/lib/mysql

volumes:
  mysql_data:
```

---

4. Post-Deployment Checklist

Security

- ☐ Change default passwords
- ☐ Set DEBUG=False
- ☐ Configure ALLOWED_HOSTS
- ☐ Setup SSL certificate
- ☐ Configure firewall rules

Performance

- ☐ Setup database indexing
- ☐ Configure caching (Redis/Memcached)
- ☐ Optimize static files serving
- ☐ Setup monitoring (CloudWatch/New Relic)

Monitoring

- ☐ Setup application logs
- ☐ Configure error tracking (Sentry)
- ☐ Setup uptime monitoring
- ☐ Database performance monitoring

Backup

- ☐ Database backup strategy
- ☐ Application files backup
- ☐ Automated backup scripts
- ☐ Test backup restoration

---

5. Environment-Specific Settings

Production Settings

```
# settings/production.py
from .base import *

DEBUG = False
ALLOWED_HOSTS = ['yourdomain.com', 'www.yourdomain.com']

# Security settings
SECURE_SSL_REDIRECT = True
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
SESSION_COOKIE_SECURE = True
CSRF_COOKIE_SECURE = True
```

```python
# Database
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': os.getenv('DATABASE_NAME'),
        'USER': os.getenv('DATABASE_USER'),
        'PASSWORD': os.getenv('DATABASE_PASSWORD'),
        'HOST': os.getenv('DATABASE_HOST'),
        'PORT': os.getenv('DATABASE_PORT', '3306'),
        'OPTIONS': {
            'charset': 'utf8mb4',
        }
    }
}

# Static files
STATIC_ROOT = '/var/www/static/'
```

---

6. Troubleshooting

Common Issues

Static Files Not Loading

```bash
# Collect static files
python manage.py collectstatic --noinput

# Check nginx static files configuration
sudo nginx -t
```

Database Connection Error

```bash
# Test database connection
python manage.py dbshell

# Check environment variables
python manage.py shell -c "from django.conf import settings; print(settings.DATABASES)"
```

Permission Errors

```bash
# Fix file permissions
sudo chown -R ubuntu:ubuntu /home/ubuntu/travel-booking-app/
sudo chmod -R 755 /home/ubuntu/travel-booking-app/
```

Logs and Debugging

```
# Application logs
sudo journalctl -u travel-booking -f

# Nginx logs
sudo tail -f /var/log/nginx/error.log
sudo tail -f /var/log/nginx/access.log

# Django logs
tail -f /home/ubuntu/travel-booking-app/logs/django.log
```

Support

For deployment issues: 1. Check application logs 2. Verify environment variables 3. Test database connectivity 4. Check security group/firewall rules 5. Review nginx/apache configuration

Happy Deploying!