Deploying a Django Project on Railway.com

✅ Complete Guide to Deploying a Django Project on Railway.com

This guide documents every step required to successfully deploy a Django project on Railway.com, based on real deployment experience.

📦 1. Initial Project Setup

Create Django Project

```
django-admin startproject your_project_name
cd your_project_name
```

Create a Django App

```
python manage.py startapp your_app_name
```

🧩 2. Install Required Packages

Add these to `requirements.txt` :

```
asgiref==3.8.1
dj-database-url==2.3.0
Django==5.2
django-admin-honeypot-updated-2021==1.2.0
django-environ==0.12.0
django-ipware==7.0.1
djangorestframework==3.16.0
gunicorn==23.0.0
packaging==24.2
pillow==11.1.0
```

```
psycopg2-binary==2.9.10
python-ipware==3.0.0
sqlparse==0.5.3
typing_extensions==4.13.2
whitenoise==6.9.0
```

Then install:

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

📁 3. Project Configuration

Add Environment Support in `settings.py`

```
from pathlib import Path
import os
import environ
# Initialize environ
env = environ.Env()
environ.Env.read_env(os.path.join(BASE_DIR, '.env'))  # ✅ Load .env using full path
ENVIRONMENT = env('ENVIRONMENT', default='production')
```

Database Setup in `settings.py`

```
if ENVIRONMENT == 'development':
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.sqlite3',
            'NAME': BASE_DIR / 'db.sqlite3',
        }
    }
else:
   DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.postgresql',
            'NAME': env('DB_NAME'),
            'USER': env('DB_USER'),
            'PASSWORD': env('DB_PASSWORD'),
            'HOST': env('DB_HOST'),
            'PORT': env('DB_PORT'),
        }
    }
```

Static Files Configuration

```
STATIC_URL = 'static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')

STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'
MIDDLEWARE.insert(1, 'whitenoise.middleware.WhiteNoiseMiddleware')
```

🔐 4. Email SMTP Setup (Optional)

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_HOST_USER = env('EMAIL_HOST_USER')
EMAIL_HOST_PASSWORD = env('EMAIL_HOST_PASSWORD')
```

📄 5. Create `.env` File (local and Railway)

```
ENVIRONMENT=development

# Django Secret Key (for development use only; change this in production)
SECRET_KEY=your django secret key

# Email (Gmail SMTP)
EMAIL_HOST_USER=your host email
```

```
EMAIL_HOST_PASSWORD=app password not gmail or email  # App password, not your Gmail login password

# PostgreSQL database (used by dj_database_url in production)
# DATABASE_URL=postgresql://postgres:RBJNrwiBTYOxMVmJOMRRPDYOzQGBlcMs@postgres.railway.internal:5432/railway
DATABASE_URL=your postgreSQL url from railway.com's postgreSQL

# Optional: Explicit DB info (useful for other tools or debugging, not used by default in Django settings)
DB_NAME=railway
DB_USER=postgres
DB_PASSWORD=your db password
DB_HOST=your db host
DB_PORT=your db port
#all information you'll get from settings.py and railway.com postgreSQL
```

> Don't forget to add .env to .gitignore

🚀 6. Prepare for Deployment

Create `Procfile` (no extension)

```
web: gunicorn your_project_name.wsgi
```

Make Migrations

```
python manage.py makemigrations
python manage.py migrate
```

Create Superuser

```
python manage.py createsuperuser
```

Collect Static Files

```
python manage.py collectstatic
```

📤 7. Push to GitHub

```
git init
git add .
git commit -m "Initial commit"
git remote add origin <https://github.com/yourusername/yourrepo.git>
git push -u origin main
```

☁️ 8. Deploy to Railway

1. Go to https://railway.app
2. Click **New Project > Deploy from GitHub Repo**
3. Select your repo
4. Railway will auto-detect the project
5. Add environment variables from your `.env` file
6. Railway auto-provisions PostgreSQL if selected
    a. if not auto selected then follow bellow steps:
        i. Click New project > Deploy postgre SQL
        ii. then go variables
            1. Copy all variables and value then
            2. command chatgpt or manually set postgre SQL database schema like (8)
            3. then update .env according to
7. **Set Environment Variables on Railway**
    a. Go to your project on Railway.
    b. Click on the **“Settings”** tab.

c. Scroll down to **“Environment”** or **“Variables”**.

d. Add the following variables (you must match these exactly with what your code expects):

Key	**Value**
SECRET_KEY	your_django_secret_key
ENVIRONMENT	production
DB_NAME	your_database_name
DB_USER	your_database_user
DB_PASSWORD	your_database_password
DB_HOST	your_postgres_host (e.g. pg.up.railway.app)
DB_PORT	usually 5432
EMAIL_HOST_USER	your_email_address
EMAIL_HOST_PASSWORD	your_email_app_password

> 💡 🔐 If you’re using Railway’s built-in PostgreSQL plugin, click on the plugin and Railway will show all the database credentials.

a. Then changed and deplooy

8. Copy PostgreSQL URL and set in `.env` as `DATABASE_URL`
9. Manually replace database settings in `settings.py` (if needed)

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'your_db_name',
        'USER': 'your_user',
        'PASSWORD': 'your_password',
        'HOST': 'your_host',
        'PORT': 'your_port',
    }
}
```

> Tip: Railway shows actual HOST and PORT under the PostgreSQL plugin tab

10. update .env for above database
11. then push

✅ 9. Final Touches

- Add allowed host and CSRF settings:

```
ALLOWED_HOSTS = ['your-app-name.up.railway.app']
CSRF_TRUSTED_ORIGINS = ['<https://your-app-name.up.railway.app>']
```

- Enable HTTPS security:

```
CSRF_COOKIE_SECURE = True
SESSION_COOKIE_SECURE = True
#if shows infinity loop or url does not work then bellow line must be comment out
SECURE_SSL_REDIRECT = True
```

🌐 10. Access Your App

Visit your Railway app link: `https://your-app-name.up.railway.app`

Admin Panel: `https://your-app-name.up.railway.app/admin`

💡 Tips & Troubleshooting

- Make sure `.env` is complete and accurate
- Double-check PostgreSQL host & port from Railway dashboard
- Use `railway logs` for debugging if container fails
- Confirm `Procfile` has no file extension

🎉 Done!

Your Django project is now live on Railway 🚂✨