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`:

```
django
dj-database-url
django-environ
gunicorn
psycopg2-binary
whitenoise
```

Then install:

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

📁 3. Project Configuration

Add Environment Support in `settings.py`

```
import dj_database_url
from environ import Env

env = Env()
env.read_env()

ENVIRONMENT = env('ENVIRONMENT', default='production')
SECRET_KEY = env('SECRET_KEY')
```

Database Setup in `settings.py`

```
if ENVIRONMENT == 'development':
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.sqlite3',
            'NAME': BASE_DIR / 'db.sqlite3',
        }
    }
else:
    DATABASES = {
        'default': dj_database_url.config(conn_max_age=600, ssl_require=Tru
e)
    }
```

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.CompressedManifestStaticFi
lesStorage'
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=production
SECRET_KEY=your_django_secret_key
EMAIL_HOST_USER=your_email@gmail.com
EMAIL_HOST_PASSWORD=your_gmail_app_password
```

> 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
7. Copy PostgreSQL URL and set in `.env` as `DATABASE_URL`
8. 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

---

✅ 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
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 🚂✨