Django Docker-Ready Project Structure

Containerized Django with Docker Compose, Gunicorn, and Nginx. Ready for production deployment.

#django #python #docker #deployment #backend

PNG PDF Copy Prompt

Project Directory

```
myproject/
    manage.py
  config/
    __init__.py
  settings/
      __init__.py
      base.py
      local.py
      production.py  Docker-specific…
    urls.py
    wsgi.py
    asgi.py
  apps/
  core/
      __init__.py
      apps.py
      models.py
      views.py
      urls.py
    migrations/
        __init__.py
    __init__.py
  docker/  Docker configur…
  django/
      Dockerfile  Multi-stage bui…
      entrypoint.sh  Wait for DB, mi…
      start.sh  Gunicorn startup
    nginx/
      Dockerfile
      nginx.conf  Reverse proxy c…
      default.conf  Server block
    postgres/
      init.sql  Initial DB setup
  scripts/  Helper scripts
    wait-for-it.sh  Wait for service
    backup-db.sh
    restore-db.sh
  templates/
    base.html
  static/
    css/
    js/
  staticfiles/  collectstatic o…
  media/  User uploads (v…
  requirements/
    base.txt
    local.txt
    production.txt  gunicorn, psyco…
  docker-compose.yml  Development sta…
  docker-compose.prod.yml  Production over…
  .env.example
  .env.docker  Docker-specific…
  .dockerignore
  .gitignore
  Makefile  Common commands
```

Why This Structure?

This structure separates Docker configs into `docker/` with per-service subdirectories. Multi-stage Dockerfile keeps images small. Separate compose files for dev vs prod. Scripts folder for common operations like DB backups.

Key Directories

`docker/django/` - Dockerfile, entrypoint, Gunicorn start script

`docker/nginx/` - Nginx reverse proxy configuration

`scripts/` - Helper scripts (wait-for-it, backups)

`staticfiles/` - collectstatic output, served by Nginx

Docker Services

`django` - Gunicorn serving the app on port 8000

`nginx` - Reverse proxy, static files, SSL termination

`postgres` - PostgreSQL database with persistent volume

`redis` - Cache and session store (optional)

Getting Started

1. `cp .env.example .env && cp .env.docker.example .env.docker`
2. `docker-compose build`
3. `docker-compose up -d`
4. `docker-compose exec django python manage.py migrate`
5. `docker-compose exec django python manage.py createsuperuser`
6. Visit `http://localhost`

Dockerfile Stages

`base` - Python, system deps, create user

`builder` - Install Python packages to wheels

`production` - Copy wheels, minimal runtime image

Production Settings

- `DEBUG=False`, `ALLOWED_HOSTS` from env
- Database from `DATABASE_URL` env var
- Static files served by Nginx (not Django)
- `SECRET_KEY` from environment
- `SECURE_SSL_REDIRECT`, `CSRF_COOKIE_SECURE` enabled

Trade-offs

`Learning curve` - Docker adds complexity for newcomers

`Local dev speed` - Container rebuilds slower than native

`Resource usage` - Multiple containers need more RAM