FastAPI Docker-Ready Project Structure

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

#fastapi #python #api #docker #deployment #gunicorn #nginx

PNG PDF Copy Prompt

Project Directory

```
myproject/
    main.py  Uvicorn entry p…
  app/  Application code
    __init__.py
    config.py  Pydantic settin…
  core/
      __init__.py
      database.py  Async SQLAlchemy
      security.py
  models/
      __init__.py
      base.py
      user.py
  schemas/
      __init__.py
      user.py
  api/
      __init__.py
    v1/
        __init__.py
        router.py
        health.py  Liveness/readin…
docker/  Docker configur…
  app/
      Dockerfile  Multi-stage bui…
      entrypoint.sh  Wait for DB, ru…
      gunicorn.conf.py  Worker config
  nginx/
      Dockerfile
      nginx.conf  Reverse proxy c…
      default.conf  Server block
scripts/  Helper scripts
    wait-for-it.sh  Wait for DB ava…
    run-migrations.sh
alembic/
    env.py
    script.py.mako
  versions/
      .gitkeep
  alembic.ini
tests/
    __init__.py
    conftest.py
    test_health.py  Smoke tests for…
docker-compose.yml  Local dev envir…
docker-compose.prod.yml  Production over…
requirements.txt
requirements-dev.txt
.env.example
.env.docker  Docker-specific…
.dockerignore
.gitignore
Makefile  Common commands
```

Why This Structure?

This structure is optimized for containerized deployments. Gunicorn with Uvicorn workers handles production traffic, Nginx provides reverse proxy and static file serving, and Docker Compose orchestrates the stack. Health endpoints enable proper orchestration integration.

Key Directories

docker/app/ - FastAPI container (Dockerfile, entrypoint, Gunicorn config)

docker/nginx/ - Reverse proxy container

scripts/ - Startup and maintenance scripts

app/api/v1/health.py - Kubernetes-ready health probes

Getting Started

1. `cp .env.example .env`
2. `docker-compose up --build`
3. Visit http://localhost/docs for Swagger UI
4. `docker-compose exec app alembic upgrade head`

Docker Files Explained

docker-compose.yml - Dev: hot reload, debug ports

docker-compose.prod.yml - Prod: optimized build, no debug

Dockerfile - Multi-stage: builder → runtime

gunicorn.conf.py - Worker count, timeouts, logging

Gunicorn Config

```python
# gunicorn.conf.py
import multiprocessing

# Worker processes
workers = multiprocessing.cpu_count() * 2 + 1

# Worker class (Uvicorn for async support)
worker_class = "uvicorn.workers.UvicornWorker"

# Server socket
bind = "0.0.0.0:8000"

# Worker timeout
timeout = 120
keepalive = 2

# Logging
accesslog = "-"
errorlog = "-"
loglevel = "info"

# Process naming
proc_name = "fastapi_app"

# Max requests per worker
max_requests = 1000
max_requests_jitter = 100
```

Health Endpoints

/health/live - Liveness: returns 200 if process is running

/health/ready - Readiness: returns 200 if can connect to DB

/health/startup - Startup: for slow initialization (optional)

When To Use This

- Deploying to Docker/Kubernetes environments
- CI/CD pipelines requiring containerized builds
- Teams wanting reproducible environments
- Production deployments with load balancing
- Microservice architectures

Trade-offs

Local dev complexity - Docker adds startup time vs. bare uvicorn

More configuration - Nginx, Gunicorn, and Docker configs to maintain

Resource usage - Containers use more memory than direct execution