Deployment Guide

Drop Deployment Guide

Last updated: 2026-03-03 **Source:** `src/drop-app/Dockerfile`, `docker-compose.yml`, `DOCKER.md`

> **NOTE (2026-03-03):** This document was updated for ADR-014 (PostgreSQL-only). The SQLite single-container deployment and `better-sqlite3` native dependency have been removed. Current deployment: Docker + PostgreSQL 16 (dev), AWS App Runner + RDS (production).

Architecture Overview

Drop uses a **multi-stage Docker build** producing a minimal Node.js 22 Alpine production image. The application is a Next.js 16 standalone server.

Build stages (from `Dockerfile:1-41`):

Stage	Base	Purpose
`deps`	`node:22-alpine`	Install `node_modules` via `npm ci`.
`builder`	`node:22-alpine`	Copy deps + source, run `npm run build` (Next.js standalone output).
`runner`	`node:22-alpine`	Minimal production image. Copies only `public/`, `.next/standalone/`, `.next/static/`.

Security features in the runner stage (`Dockerfile:25-26`):

- Non-root user: `nextjs` (UID 1001, GID 1001)
- Data directory `/app/data` owned by `nextjs:nodejs`
- No build tools or source code in production image

Deployment Configurations

1. Local Development -- `docker-compose.yml`

PostgreSQL 16 + Drop app (ADR-014).

File: `src/drop-app/docker-compose.yml:1-22`

```
services:
  drop-app:
    build: .
    ports:
      - "3000:3000"
    environment:
      - JWT_SECRET=${JWT_SECRET:?JWT_SECRET is required}
      - NODE_ENV=production
      - NEXT_PUBLIC_SERVICE_MODE=mock
    volumes:
      - drop_data:/app/data
    healthcheck:
      test: ["CMD", "wget", "--no-verbose", "--tries=1", "--spider",
"http://localhost:3000/api/health"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 10s
    restart: unless-stopped
```

Quick start:

```
export JWT_SECRET="your-secure-random-string-min-32-chars"
docker compose up -d
```

Data persistence: PostgreSQL data stored in Docker volume `drop_pgdata`.

2. Production (PostgreSQL) -- `docker-compose.production.yml`

Multi-container setup with separate PostgreSQL 16 database.

File: `src/drop-app/docker-compose.production.yml:1-38`

```
services:
  drop-app:
    build: .
    ports:
      - "3000:3000"
    depends_on:
      postgres:
        condition: service_healthy
    restart: unless-stopped

  postgres:
    image: postgres:16-alpine
    environment:
      - POSTGRES_DB=drop
      - POSTGRES_USER=drop
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD:-drop_local_dev}
    volumes:
      - postgres_data:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U drop"]
      interval: 10s
      timeout: 5s
      retries: 5
```

Quick start:

```
export JWT_SECRET="your-secure-random-string-min-32-chars"
export POSTGRES_PASSWORD="secure-postgres-password"
docker compose -f docker-compose.production.yml up -d
```

3. Fly.io Staging -- `fly.toml`

File: `src/drop-app/fly.toml:1-28`

Setting	Value
App name	`drop-staging`
Region	`arn` (Stockholm -- closest to Norway)

Setting	Value
Internal port	3000
Force HTTPS	`true`
Auto-stop machines	`stop` (scales to zero)
Auto-start machines	`true`
Min machines	0
Persistent storage	Volume `drop_data` mounted at `/app/data`

Health check: `GET /api/health` every 30s, 5s timeout, 10s grace period.

Environment Variables

Variable	Required	Default	Description
`JWT_SECRET`	**Yes** (production)	Dev: `process.cwd()` hash	JWT signing secret. Minimum 32 characters. Fatal error if missing in production.
`NODE_ENV`	No	`development`	Set to `production` in containers. Controls seed data gating.
`NEXT_PUBLIC_SERVICE_MODE`	No	-	Set to `mock` for MVP mode (no external API calls).
`DATABASE_URL`	**Yes**	-	PostgreSQL 16 connection string. Required in all environments. Local dev: `postgresql://drop:dev_only_not_a_secret@localhost:5433/drop_dev`
`POSTGRES_PASSWORD`	Production only	`drop_local_dev`	PostgreSQL password (production compose).
`PORT`	No	`3000`	HTTP server port.
`HOSTNAME`	No	`0.0.0.0`	Server bind address.

Database: PostgreSQL 16 is required in all environments. There is no SQLite fallback (ADR-014).

Health Check

Endpoint: `GET /api/health` **Source:** `src/drop-app/src/app/api/health/route.ts:1-35`

The health check performs a real database query (`SELECT 1 as ok`) and reports latency.

Success response (200):

```
{
  "status": "ok",
  "version": "0.1.0",
  "uptime": 123,
  "db": "connected",
  "dbLatencyMs": 5,
  "timestamp": "2026-02-13T12:00:00.000Z"
}
```

Failure response (503):

```
{
  "status": "error",
  "db": "disconnected",
  "timestamp": "..."
}
```

Building from Source

```
# Build Docker image
docker build -t drop-app .

# Run standalone container
docker run -d \
  -p 3000:3000 \
  -e JWT_SECRET="your-secret-min-32-chars" \
  -v drop_data:/app/data \
  --name drop-app \
  drop-app
```

Data Backup and Restore

Production Backups (AWS RDS)

Production database is PostgreSQL 16 on AWS RDS. Backups are managed by AWS:

- **Automated backups:** Daily snapshots, 7-day retention (configured in RDS)
- **Point-in-time recovery:** Available within the 7-day retention window
- **Manual snapshot:** Via AWS Console or CLI before major deployments

Create a manual RDS snapshot before deployments:

```
aws rds create-db-snapshot \
  --db-instance-identifier drop-production \
  --db-snapshot-identifier drop-pre-deploy-$(date +%Y%m%d-%H%M%S)
```

Restore from snapshot: Via AWS Console → RDS → Snapshots → Restore.

Local Dev Backups (Docker)

Local development data in the `drop_pgdata` Docker volume is disposable. Recreate with:

```
docker compose down -v   # Remove volume (deletes local data)
docker compose up -d
make db-push && npm run db:seed
```

Backup Verification

Verify production database connectivity and integrity:

```
# Check health endpoint
curl https://your-app-runner-url/api/health

# Connect to RDS (requires VPN or bastion)
psql $DATABASE_URL -c "SELECT COUNT(*) FROM users;"
```

Demo User

In non-production mode (`NODE_ENV !== 'production'`), a demo user is seeded:

Field	Value

Email	`amir@example.com`
Password	`demo1234`
Role	merchant

Source: Drizzle seed script in `src/shared/db/seed.ts`. Gated behind `NODE_ENV !== 'production'`.

Troubleshooting

Container won't start:

```
docker compose logs
docker compose exec drop-app env | grep JWT_SECRET
```

Database connection issues:

```
# Check PostgreSQL container is running
docker compose ps

# Test connection
docker compose exec db psql -U drop -d drop_dev -c "SELECT COUNT(*) FROM users;"

# Check app DATABASE_URL is set correctly
docker compose exec drop-app env | grep DATABASE_URL
```

Permission denied:

```
docker compose down -v   # Remove volumes
docker compose up -d     # Recreate with correct permissions
```

Cleanup:

```
docker compose down      # Stop containers
docker compose down -v   # Stop + remove volumes (WARNING: deletes data)
docker rmi drop-app      # Remove image
```

Revision #1
Created 2026-05-23 10:58:08 UTC by John
Updated 2026-05-23 10:58:08 UTC by John