Mezzanine — New Project to Production

A laptop-first checklist for getting a new app deploy-ready before it reaches Mezzanine. Every rule here exists because the failure happened in production and was fixed.

1 Decide these first

Decision	Why it matters
What port does the app listen on?	Must equal Mezzanine's Container port, or the healthcheck fails (curl error 56).
Is it stateful? (DB, uploads, any file)	Stateful apps MUST get a named volume, or data is lost on redeploy.
SQLite or a database server?	SQLite → a volume. Postgres/MySQL → provision the DB + DATABASE_URL first.
Which env vars / secrets?	They must be set before first boot (e.g. AUTH_TRUST_HOST or next-auth 500s).
Public domain + which box?	DNS → box → nginx → container port. Confirm before debugging the wrong box.

2 Prepare your repo

Resilient install — strict `npm ci` aborts the build on any lockfile drift (macOS lock vs alpine/musl build, Tailwind/eslint/sharp optional deps). Fall back to install:

```
COPY package.json package-lock.json* ./
RUN npm ci --no-audit --no-fund || npm install --no-audit --no-fund
```

.gitignore — secrets out, large media handled (GitHub rejects files over 100 MB):

```
.env
.env.*
!.env.example
*.mp4
!public/assets/hero.mp4   # < 100 MB, actually used
```

Health endpoint — expose a path that returns 200, ideally DB-backed (GET /api/health). Never assume "/" returns 200: auth/i18n apps redirect it to a 307, which fails the healthcheck.

Test the production shape locally — if it does not run on your laptop, it will not in Mezzanine:

```
docker build -t myapp .
docker run -d --name myapp -p 8080:3000 --env-file .env -v myapp-data:/app/data myapp
curl -fsS http://localhost:8080/api/health        # expect 200
# persistence proof: destroy + recreate, data MUST survive
docker rm -f myapp
docker run -d --name myapp -p 8080:3000 --env-file .env -v myapp-data:/app/data myapp
```

3 Data & persistence — the rule that matters most

A Dockerfile VOLUME does NOT persist your data. Started with "docker run" and no -v, it creates a brand-new empty anonymous volume on every launch — data gone on every deploy, old data stranded in an orphan volume a routine prune then deletes for good. **Persistence must come from the run command mounting a named volume** (-v app-data:/app/data), not the Dockerfile.

Databases: provision first → set DATABASE_URL → run migrations (prisma migrate deploy; generate only builds the client, it does not create tables). Back up before every redeploy / migration.

4 Environment variables

App type	Required env
next-auth (v5)	AUTH_TRUST_HOST=true (fixes UntrustedHost 500), AUTH_SECRET (openssl rand -base64 32), AUTH_URL=https://your-domain
Database app	DATABASE_URL — and run migrations once
Common	NODE_ENV=production, PORT, SMTP_*, etc.

All config via env, set at deploy time — never baked into the image, never committed. Commit .env.example (keys only); gitignore .env.

5 Pre-flight checklist

- docker build succeeds from a clean checkout; lockfile committed + in sync
- App reads all config from env; .env.example complete; no secrets in git
- Stateful? data dir mounted as a named volume and survives docker rm -f + recreate locally
- DB app? DATABASE_URL documented and migrations run as a separate step
- GET /api/health returns 200 without auth
- EXPOSE = listen port, noted for the Container port field
- No file over 100 MB committed; secrets only in .env (gitignored)

6 Deploy on Mezzanine

Mezzanine field	Value
Container port	your EXPOSE / listen port
Environment	every key from .env.example, with real values
Volume	app-data:/app/data (mandatory for any stateful app)
Healthcheck URL	http://127.0.0.1:<host-port>/api/health (a 200 path)

Tip: the "Start a project" generator in Mezzanine produces a starter with all of this pre-wired, plus a mezzanine.json the deploy wizard can read.

7 Do / Don’t

DO	DON’T
Mount a named volume for any data	Rely on a Dockerfile VOLUME to persist
npm ci || npm install	Bare npm ci (breaks on lock drift)
Set env + DB before the first deploy	Bake secrets into the image / commit .env
Healthcheck a 200 path	Assume "/" returns 200
Back up before redeploy / migration	docker volume prune / system prune --volumes / compose down -v (these delete databases)
Match Container port to EXPOSE	Commit files over 100 MB

8 When something breaks

Symptom	Cause → Fix
npm ci ... Missing @emnapi/... from lock file	Lockfile drift → resilient install (§2)
[auth][error] UntrustedHost 500	next-auth behind a proxy → AUTH_TRUST_HOST=true (§4)
prisma.user.count() invalid 500	No/wrong DATABASE_URL, no migrations → provision + migrate (§3)
Healthcheck fails on 307	"/" redirects → healthcheck a 200 path (§2)
App shows the setup screen after a deploy	Data went to a fresh anonymous volume → named volume (§3)
curl error 56 at healthcheck	Container port ≠ app port → match the port (§1)

Full guide: https://docs.mezzanine.cloud/guides/ship-to-production/