AI for Science – Deployment & Operations Guide (Hetzner + Plesk)

Date: 2025-09-28

# Overview

This document captures the exact, working configuration used to deploy the "ai-for-science" portfolio project on a Hetzner dedicated server running Plesk. It covers Docker installation on Ubuntu/Debian with Plesk, repository layout, Compose setup, Plesk reverse proxy configuration, path-based hosting under /ai-for-science, and a full runbook for operating and testing the system.

Environment (as deployed):  
- Domain: squeezecost.com  
- API subdomain: api.squeezecost.com  
- Frontend served under subpath: https://squeezecost.com/ai-for-science/  
- API served under path prefix: https://api.squeezecost.com/ai-for-science  
- Host path for project: /opt/ai-for-science  
- Container loopback bindings: 127.0.0.1:3001 (frontend), 127.0.0.1:8001 (backend)  
- No public container ports are exposed; Plesk terminates TLS and proxies to loopback.

# Repository structure (key parts)

/opt/ai-for-science  
├─ backend/ FastAPI app, Redis client, middlewares (timeouts/errors)  
├─ frontend/ Next.js 15 app (React 18.2), basePath-aware  
├─ deploy/ (Hetzner/Plesk assets if needed)  
├─ docker-compose.yml Base compose (no public ports)  
├─ docker-compose.override.yml Loopback port publishing for Plesk nginx  
└─ .env DOMAIN=squeezecost.com

# Docker on Ubuntu/Debian with Plesk – Installation (working solution)

Run these commands as root (or with sudo). This resolves the common Signed-By conflict that occurs when Plesk's Docker extension adds its own apt source.

1) Prerequisites  
 sudo apt-get update  
 sudo apt-get install -y ca-certificates curl gnupg lsb-release

2) Docker official GPG key  
 sudo install -m 0755 -d /etc/apt/keyrings  
 curl -fsSL https://download.docker.com/linux/$(. /etc/os-release; echo "$ID")/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  
 sudo chmod a+r /etc/apt/keyrings/docker.gpg

3) Add Docker official repo  
 echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/$(. /etc/os-release; echo "$ID") $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

4) Disable Plesk’s duplicate Docker list (prevents Signed-By conflict)  
 sudo mv /etc/apt/sources.list.d/plesk-ext-docker.list /etc/apt/sources.list.d/plesk-ext-docker.list.disabled  
 # (Optional cleanup)  
 sudo rm -f /etc/apt/sources.list.d/plesk-ext-docker.list.disabled  
 sudo rm -f /etc/apt/keyrings/plesk-ext-docker.gpg

5) Install Docker Engine + Compose plugin  
 sudo apt-get update  
 sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

6) Enable & start  
 sudo systemctl enable docker  
 sudo systemctl start docker

7) Verify  
 docker --version  
 docker compose version  
 sudo docker run --rm hello-world

Notes:  
- Plesk’s Docker extension continues to work—it uses the system Docker daemon. You’ve only disabled its duplicate apt source.  
- Keep 80/443 open in the firewall for TLS; do NOT open 3001/8001/6379 publicly.

# Compose configuration (final)

docker-compose.yml  
------------------  
services:  
 redis:  
 image: redis:7-alpine  
 restart: unless-stopped  
 volumes:  
 - redis-data:/data  
 networks: [internal]

backend:  
 build: ./backend  
 restart: unless-stopped  
 environment:  
 - ENV=prod  
 - REDIS\_URL=redis://redis:6379/0  
 - DATABASE\_URL=sqlite+aiosqlite:////data/app.db  
 - API\_ROOT\_PATH=/ai-for-science  
 volumes:  
 - backend-data:/data  
 depends\_on:  
 - redis  
 networks: [internal]

frontend:  
 build:  
 context: ./frontend  
 args:  
 BASE\_PATH: /ai-for-science  
 NEXT\_PUBLIC\_API\_URL: https://api.${DOMAIN}/ai-for-science  
 environment:  
 - NEXT\_PUBLIC\_API\_URL=https://api.${DOMAIN}/ai-for-science  
 restart: unless-stopped  
 depends\_on:  
 - backend  
 networks: [internal]

networks:  
 internal:

volumes:  
 redis-data:  
 backend-data:

docker-compose.override.yml  
---------------------------  
services:  
 backend:  
 ports: ["127.0.0.1:8001:8000"]  
 frontend:  
 ports: ["127.0.0.1:3001:3000"]

.env  
----  
DOMAIN=squeezecost.com

# Frontend Dockerfile (final)

frontend/Dockerfile  
-------------------  
# ---------- Builder ----------  
ARG BASE\_PATH  
ARG NEXT\_PUBLIC\_API\_URL

FROM node:20-bullseye AS builder

ARG BASE\_PATH  
ARG NEXT\_PUBLIC\_API\_URL

WORKDIR /app  
ENV NEXT\_TELEMETRY\_DISABLED=1  
ENV NEXT\_PUBLIC\_BASE\_PATH=${BASE\_PATH}  
ENV NEXT\_PUBLIC\_API\_URL=${NEXT\_PUBLIC\_API\_URL}

RUN apt-get update && apt-get install -y --no-install-recommends ca-certificates python3 make g++ && rm -rf /var/lib/apt/lists/\*

COPY package.json package-lock.json\* pnpm-lock.yaml\* yarn.lock\* ./  
RUN if [ -f package-lock.json ]; then npm ci; elif [ -f pnpm-lock.yaml ]; then npm install -g pnpm && pnpm i --frozen-lockfile; elif [ -f yarn.lock ]; then corepack enable && yarn install --immutable; else npm install; fi

COPY . .  
RUN mkdir -p public  
RUN npm run build

# ---------- Runtime ----------  
FROM node:20-bullseye AS runner

WORKDIR /app  
ENV NODE\_ENV=production  
ENV NEXT\_TELEMETRY\_DISABLED=1

COPY --from=builder /app/package.json ./package.json  
COPY --from=builder /app/.next ./.next  
COPY --from=builder /app/public ./public  
COPY --from=builder /app/node\_modules ./node\_modules  
COPY --from=builder /app/next.config.\* ./  
COPY --from=builder /app/tsconfig.json ./

EXPOSE 3000  
CMD ["npm", "start"]

# Next.js configuration

frontend/next.config.ts  
-----------------------  
import type { NextConfig } from "next";

const nextConfig: NextConfig = {  
 basePath: process.env.NEXT\_PUBLIC\_BASE\_PATH || "/ai-for-science",  
 trailingSlash: true,  
};

export default nextConfig;

package.json (key versions)  
---------------------------  
- next: 15.0.0  
- react: 18.2.0  
- react-dom: 18.2.0  
- typescript: ^5.5.x  
- @types/node, @types/react, @types/react-dom installed as devDependencies

# Plesk nginx configuration (reverse proxy)

squeezecost.com (UI under subpath)  
----------------------------------  
Plesk → Domains → squeezecost.com → Apache & nginx Settings → Additional Nginx directives

location = /ai-for-science {  
 return 301 /ai-for-science/;  
}

location ^~ /ai-for-science/ {  
 proxy\_pass http://127.0.0.1:3001;  
 proxy\_http\_version 1.1;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
}

api.squeezecost.com (API under path prefix)  
-------------------------------------------  
Plesk → Domains → api.squeezecost.com → Apache & nginx Settings → Additional Nginx directives

location ^~ /ai-for-science/ {  
 proxy\_pass http://127.0.0.1:8001;  
 proxy\_http\_version 1.1;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
}

TLS  
---  
Issue Let's Encrypt certificates for both domains in Plesk before testing.

# Runbook – start/stop/rebuild

From /opt/ai-for-science:

# Start or restart everything (keeps volumes/data)  
docker compose -f docker-compose.yml -f docker-compose.override.yml up -d --build

# Stop stack  
docker compose -f docker-compose.yml -f docker-compose.override.yml down

# Status and logs  
docker compose -f docker-compose.yml -f docker-compose.override.yml ps  
docker compose -f docker-compose.yml -f docker-compose.override.yml logs -f backend  
docker compose -f docker-compose.yml -f docker-compose.override.yml logs -f frontend

# Rebuild only frontend after code/lockfile changes  
docker compose -f docker-compose.yml -f docker-compose.override.yml build --no-cache frontend  
docker compose -f docker-compose.yml -f docker-compose.override.yml up -d frontend

# Rebuild only backend  
docker compose -f docker-compose.yml -f docker-compose.override.yml build backend  
docker compose -f docker-compose.yml -f docker-compose.override.yml up -d backend

# Testing checklist

Local (bypass Plesk):  
- API health:  
 curl -sS http://127.0.0.1:8001/api/v1/healthz  
- Generate call (reverse text demo):  
 curl -sS -X POST http://127.0.0.1:8001/api/v1/llm/generate -H 'Content-Type: application/json' -d '{"prompt":"hello world"}'  
- Frontend base path:  
 curl -I http://127.0.0.1:3001/ai-for-science/

Public (through Plesk + TLS):  
- API health:  
 curl -sS https://api.squeezecost.com/ai-for-science/api/v1/healthz  
- Generate call:  
 curl -sS -X POST https://api.squeezecost.com/ai-for-science/api/v1/llm/generate -H 'Content-Type: application/json' -d '{"prompt":"hello world"}'  
- Frontend:  
 https://squeezecost.com/ai-for-science/

Optional load smoke:  
 API\_URL=https://api.squeezecost.com/ai-for-science k6 run backend/k6/smoke.js

# Troubleshooting

- 404 at /ai-for-science:  
 \* Ensure Next is built with basePath (/ai-for-science). Rebuild frontend with --no-cache.  
 \* Add Nginx exact-match redirect: location = /ai-for-science { return 301 /ai-for-science/; }

- 301/308 loop between /ai-for-science and /ai-for-science/:  
 \* Add the exact-match rule above, or set trailingSlash: true in next.config.ts and rebuild.

- API 502 or 404 via public URL:  
 \* Verify Plesk Nginx directives under api.squeezecost.com use proxy\_pass http://127.0.0.1:8001 for /ai-for-science/.  
 \* Check domain has a valid certificate; review domain logs in Plesk.

- Frontend calling wrong API URL:  
 \* Confirm baked env in container:  
 docker compose -f docker-compose.yml -f docker-compose.override.yml exec frontend env | grep NEXT\_PUBLIC\_API\_URL  
 \* If incorrect, update compose build args and rebuild frontend with --no-cache.

- Docker repo Signed-By conflict:  
 \* Disable Plesk’s apt list: mv /etc/apt/sources.list.d/plesk-ext-docker.list ...disabled  
 \* Keep only the official Docker apt source with key at /etc/apt/keyrings/docker.gpg.

# Security notes

- Do not expose 3001/8001/6379 publicly; keep loopback-only bindings.  
- Use Let's Encrypt in Plesk; force HTTPS (HSTS if appropriate).  
- Backend has JSON error normalization + timeouts; add rate limiting and auth as you move beyond Step 1.  
- Move DATABASE\_URL to a managed DB (Azure SQL/Postgres) for production; SQLite is for demo/dev only.

# Next steps (Step 2 preview)

- Quality gates: ruff + mypy (backend), pytest, ESLint + TypeScript (frontend), CI workflows.  
- Observability: structured logs + OpenTelemetry, Application Insights, RED dashboards.  
- Azure Web Apps deployment (slots), GitHub Actions CD with smoke tests.  
- Stripe + Clerk integration, idempotent webhooks, and runbooks.