DEPLOYMENT & OPERATIONS MANUAL

PROJECT: SYMBIOSE GEOSPATIAL TERMINAL

Next.js 15 — NestJS — PostGIS — Python LiDAR Engine

Component:	Full-Stack Geospatial Application
Frontend:	Next.js 15 (App Router) + Mapbox GL JS
Backend:	NestJS + TypeORM + Python 3 (Rasterio)
Database:	PostgreSQL 15 + PostGIS Extension
Target Env:	Docker Compose (Local / Prod)
Data Engine:	Stream-JSON ETL + Promise.all Spatial Query

Confidential - Dongshan ZHU

Contents

1 Architecture Overview

The Symbiose Geospatial Terminal is a high-performance, containerized application designed to process and visualize massive spatial datasets (BD Forêt, Etalab Cadastre, and IGN LiDAR HD).

- **Frontend:** Next.js serving a Mapbox GL JS map with dynamic viewport bounding-box guards (debounced) to prevent browser memory overload.
- **Backend:** NestJS providing RESTful endpoints. Utilizes parallel `Promise.all` execution to concurrently fetch PostGIS `ST_Intersects` data (Bio-Data & Cadastre) while spawning a Python child process for LiDAR masking.
- **ETL Pipeline:** A memory-efficient stream processing architecture (`stream-json`) engineered to handle ¿400k geo-features with a constant memory footprint (¡100MB).

2 Data Preparation & Injection

⚠ Critical Data Warning

Spatial datasets (Shapefiles, GeoJSON, GeoTIFF) scale up to several Gigabytes. **Never** commit the `data/` directory to version control. Download them manually to the host machine before building the containers.

2.1 Directory Structure Setup

Ensure the following exact structure exists at the project root:

```
mkdir -p data/BDV2 data/CPL data/lidar
```

1. **BD Forêt (Core):** Extract IGN shapefiles (`.shp`, `.dbf`, `.prj`) into `./data/BDV2/`.
2. **Cadastre (Bonus A):** Place Etalab Cadastre files (e.g., `cadastre-75-parcelles.json`) into `./data/CPL/`.
3. **LiDAR CHM (Bonus B):** Place IGNF MNH GeoTIFFs (`.tif`) into `./data/lidar/`.

3 Environment Configuration

Configure these variables in your `.env` file at the project root or inject them via your CI/CD pipeline:

Variable Name	Description
`NEXT_PUBLIC_MAPBOX_ACCESS_TOKEN`	**Required.** Mapbox API Token for rendering the base map.
`DATABASE_HOST`	Defaults to `db` in Docker Compose network.
`JWT_SECRET`	Secret key for securing API sessions and state persistence.

4 Docker Compose Deployment (Local & VPS)

This project utilizes a Monorepo Docker orchestration. The backend container seamlessly integrates the Node.js runtime with Python/GDAL bindings.

4.1 Build and Spin Up Services

```
# Build images and start services in detached mode
docker-compose up --build -d
```

4.2 Execute the ETL Seeding Pipeline

Once the containers are healthy, you must ingest the raw files into PostGIS. This script handles EPSG:2154 to EPSG:4326 reprojection, encoding fixes, and spatial indexing.

```
# Trigger the high-performance stream pipeline
docker exec -it nest_backend pnpm run seed:forest
```

4.3 Access Points

- **Geospatial Terminal UI:** `http://localhost:3001`
- **Backend REST API (Swagger):** `http://localhost:3000/api`

5 Remote Production Roadmap

For a full production launch, the following architecture is recommended:

- **Frontend Hosting (Vercel):** Zero-config deployment. Ensures global CDN distribution for static assets and Mapbox tiles.

- **Backend App (Render / Railway):** Node.js Web Service. *Crucial:* Must attach a persistent Disk Volume to store LiDAR GeoTIFFs to bypass container ephemeral storage limits.
- **Database (Supabase / Render PostgreSQL):** Managed PostgreSQL instance with the `PostGIS` extension explicitly enabled (`CREATE EXTENSION postgis;`).

6 Troubleshooting & Optimization

Symptom	Resolution
OOM (Out of Memory) during ETL	If processing departments ¿ 400k parcels (e.g., Dept 77), ensure the `stream-json` backpressure pipeline is active. Limit initial seeding to Dept 75 & 92 if host RAM is ¡ 4GB.
Map Renders as Blank Grid	Invalid or missing `NEXT_PUBLIC_MAPBOX_ACCESS_TOKEN`. Rebuild the frontend container without cache.
No Cadastre Data (Bonus A)	Ensure zoom level is ≥ 13. Verify the frontend payload is hitting the backend `ST_Intersects` concurrent logic.

End of Document