## 1 What I actually built

### End-to-end mini-pipeline for Cyprus road data

1. **Bootstrapped a geospatial lab in WSL2** – Docker-ised PostGIS 15 + 3.4, loaded the Cyprus OSM PBF with `osm2pgsql`, and enabled PostGIS / pgRouting / hstore in a clean `osm` DB.
2. **Explored the raw network and produced the first metric in < 4 h** –
   - pulled ~153 k road geometries into GeoPandas,
   - generated an H3 resolution-8 hex grid (~0.6 km$^2$ per cell),
   - spatial-joined roads➜hexes and aggregated centre-line kilometres per cell,
   - sanity-plotted the result.
3. **Re-imported with a Lua *flex* config tuned for roads only** – kept just four columns (`way_id`, `highway`, `name`, `maxspeed`, `geom`), cutting disk/RAM and ingest time dramatically.
4. **Proved the schema is fast** – added targeted B-tree, trigram and SP-GiST indexes and showed 10–100× query speed-ups with `EXPLAIN ANALYZE`.
5. **Enabled minute-diff replication** – `osm2pgsql-replication init / update` now pulls continuous changes; replication state is stored in `osm2pgsql_properties`.
6. **Instrumented Postgres** – started the container with `pg_stat_statements`, scripted resets and top-N reports to profile heavy SQL during updates.
7. **Exported lightweight GeoJSON layers** – roads and the hex choropleth leave PostGIS in EPSG 4326 so downstream tools need zero reprojection.
8. **Built vector tiles** – Tippecanoe ➜ MBTiles, then converted to single-file **PMTiles** with `pmtiles convert`; zoom 6–14 covers island-wide overview down to street level.
9. **Served and rendered them** –
   - local file server with `go-pmtiles serve …` *or* static HTTP range reads (both options),
   - a minimal MapLibre-GL page loads tiles in one line (`url:"pmtiles://…"`) and styles by `highway` class.
10. **Next-week deep-dive (tentative)** – focus on routing and BI:
    - **OSRM**
      - Compile Cyprus car profile, pre-process with `osrm-extract` & `osrm-partition`.

        - Serve `/route`, `/table` and `/nearest` endpoints; script a latency/throughput benchmark.
        - Export daily origin-destination JSON of real-world queries for later BI.
    - **Superset**
        - Connect directly to Postgres (`osm` DB) and an OSRM logs table.
        - Build dashboards:
            - heat-map of routing requests per H3 cell;
            - percentile latency charts broken down by time-of-day;
            - “missing roads” detector – hexes with high requests but zero road length.
        - Enable row-level security so future users can filter by region.

---

## 2 Tooling in one breath (and the bumps)

| Tool / lib | Role | Gotcha fixed / status |
|---|---|---|
| **Docker** + `postgis/postgis:15-3.4` | disposable PostGIS | set correct group perms in WSL2 to avoid `sudo` |
| **osm2pgsql 1.11** | initial import & diff replay | required `flat-nodes` + `--output=flex` to keep disk low |
| **Python 3.10** + **GeoPandas** + **H3 4.x** | metric prototyping | H3 v4 changed APIs – rewrote `h3.polyfill` call |
| **GDAL / ogr2ogr** | data export | Parquet driver still missing – left Parquet export TODO |
| **Tippecanoe** + **go-pmtiles** | vector tile build/serve | used external `pmtiles convert` because in-built flag did not work for some reason |
| **MapLibre GL** | browser renderer | worked out-of-the-box once tiles served |

Time sinks so far: WSL2 file-system perms, GDAL Parquet build, finding a stable PMTiles conversion path, and serving ranges of tiles.

---

## 3 Glossary of the less-obvious acronyms

- **GDAL** – Geospatial Data Abstraction Library: Swiss-army knife for geo formats.
- **H3** – Uber's hierarchical hexagonal spatial index; each cell ID is a 64-bit int.
- **GiST / SP-GiST** – Generalised search trees for spatial indexing in Postgres; SP-GiST is flatter ➔ faster for point/line bboxes.
- **EPSG / CRS** – Code list of coordinate systems; 4326 = WGS-84 lat/lon, 3857 = Web-Mercator metres. So, "EPSG : 4326," = "use the WGS-84 CRS identified by code 4326 in the EPSG database."
    - **EPSG**: *European Petroleum Survey Group* (now the IOGP "EPSG" dataset). The industry-standard catalogue of codes (e.g. *4326*, *3857*) that uniquely identify coordinate reference systems and the math to transform between them.
        - **IOGP**: *International Association of Oil & Gas Producers*
    - **CRS**: *Coordinate Reference System*. A full specification of how spatial coordinates relate to the real world—includes the datum (origin/shape of the Earth) plus the map-projection or ellipsoidal rules used.
    - **WGS**: *World Geodetic System*. A series of global datums maintained by the U.S. DoD; WGS-84 is the current version and forms the basis of GPS coordinates as well as EPSG : 4326.
- **PMTiles** – Single-file, HTTP-range-friendly container for Mapbox Vector Tiles.
- **OSRM** – Open Source Routing Machine; turns road graphs into millisecond-latency routes.

---

## 4 Logical ways to extend this

1. **Go from island to continent** – swap Cyprus PBF for Europe, scale `osm2pgsql` cache & flat-nodes, and store H3 metrics in DuckDB or BigQuery.
2. **Multiple layers** – repeat the flex-import trick for POIs, elevation contours, land/sea mask; toggle layers in MapLibre.
3. **Real-time freshness** – cron-hourly `osm2pgsql-replication update`; regenerate only *touched* hexes and re-cut PMTiles atomically.
4. **Routing & ETA analytics** – OSRM already staged; log `/route` latency, aggregate per H3 cell, and visualise in Superset.
5. **Cloud move** – bucket-host PMTiles, cache with a CDN, and migrate PostGIS to a managed service when ready.
6. **User-facing API** – wrap hex metrics and OSRM results in a FastAPI micro-service; H3 cell key makes joins trivial.
7. **Data-quality loop** – flag hexes with zero road length but many OSRM *nearest* hits; push to OSM editors.