

SA_week1_tiling

June 3, 2025

1 Data prep

1.0.1 (GDAL on legacy Ubuntu 22.04)

If no `ogr2ogr`, install GDAL: `sudo apt install gdal-bin`

Parquet support in Geospatial Data Abstraction Library (GDAL) is not available on Ubuntu 22.04, specifically `libgdal-arrow-parquet` is missing.

So this does not really help:

```
# For legacy Ubuntu 22.04:
# sudo add-apt-repository ppa:ubuntugis/ubuntugis-unstable
# sudo apt update
# sudo apt install gdal-bin libgdal-dev # pulls in GDAL 3.8+ with Arrow support
# ogrinfo --formats | grep -i parquet
# → "PARQUET -raster,vector- (rw+v): Parquet"
# THEN a whole bunch of apt update, including
# sudo apt install python-is-python3 # adds /usr/bin/python -> python3
# sudo apt install python3-venv
# (and re-install of the Python libs from 00_Setup)
# and also:
# pip install pyarrow
# DON'T forget to run
# source venv/bin/activate
# in every new terminal with Python, e.g. before
# jupyter lab
```

1.1 Export the roads layer

```
ogr2ogr -f GeoJSON roads.geojson \  
  "PG:dbname=osm host=127.0.0.1 user=postgres password=geo" \  
  -nln roads \  
  -sql "SELECT ST_Transform(geom,4326) AS geom,  
        highway,  
        name  
  FROM roads  
  WHERE highway NOT IN ('footway','path','cycleway')"
```

1.2 Export the choropleth hexes

1.2.1 Prepare the hex coverage parquet

```
[7]: import h3, sys
print(sys.version)
print(h3.versions()) # {'c': '4.1.0', 'python': '4.1.1'} + confirms you're on
↳ 4.x
```

3.10.12 (main, Feb 4 2025, 14:57:36) [GCC 11.4.0]
{'c': '4.2.1', 'python': '4.2.2'}

```
[4]: import geopandas as gpd, sqlalchemy as sa, h3
from shapely.geometry import Polygon, box

PG_DSN = "postgresql+psycopg://postgres:geo@localhost:5432/osm"
BBOX = box(32.25, 34.5, 34.7, 35.7) # Cyprus rough extent
RES = 8 # H3 resolution, 0.6 km² hexes
OUT = "cyprus_res8_roadkm.parquet"

engine = sa.create_engine(PG_DSN)

# The roads we need (SQL → GeoPandas)
roads = gpd.read_postgis("""
    SELECT ST_Transform(geom,4326) AS geom,      -- lat/lon for sjoin & tiles
           highway,
           name,
           ST_Length(geom)/1000.0 AS length_km  -- metres → km (EPSG 3857)
    FROM   roads
    WHERE  highway NOT IN ('footway','path','cycleway');
""", engine, geom_col="geom", crs="EPSG:4326")

# H3 resolution-8 grid that covers Cyprus
# h3 v3:
hex_ids = list(h3.polyfill_geojson(BBOX.__geo_interface__, RES))
hex_geom = [Polygon(h3.h3_to_geo_boundary(h, geo_json=True)) for h in hex_ids]
# H3 resolution-8 grid that covers Cyprus
# v4: build geometries without the deleted kw-arg
hex_ids = list(h3.geo_to_cells(BBOX, RES))

def to_polygon(cell):
    # returns [(lat, lng), ...] - swap to (lng, lat)
    latlngs = h3.cell_to_boundary(cell)
    return Polygon([(lng, lat) for lat, lng in latlngs])

hex_geom = [to_polygon(h) for h in hex_ids]
```

```

hex_gdf = gpd.GeoDataFrame({'h3':hex_ids, 'geometry':hex_geom}, crs="EPSG:
↳4326")

# Spatial-join roads + hex cells and aggregate km per cell
joined = gpd.sjoin(roads, hex_gdf, how="inner", predicate="intersects")
coverage = (joined.groupby('h3')           # all road segments that hit the same
↳cell
                .length_km.sum()          # add up their centre-line km (length
↳only, regardless of how wide)
                .reset_index())

# Attach the metric back to the full grid (so "road-less" hexes show 0 km)
hex_cov = (hex_gdf.merge(coverage, on='h3', how='left')
            .fillna({'length_km':0}))

# Plot for testing
# hex_cov.plot(column='length_km', cmap='viridis',
#               linewidth=0, figsize=(20,20), legend=False)

hex_cov.to_parquet(OUT, index=False)
print(f"[ok] wrote {OUT} with {len(hex_cov):,} hexes")

```

[ok] wrote cyprus_res8_roadkm.parquet with 35,447 hexes

Potential refinements

refinement	one-liner / hint
True metric length	Replace ST_Length(geom) by ST_Length(ST_Transform(geom,32636)) (UTM 36 N) for sub-1 % accuracy in Cyprus
Incremental refresh	Pass only <i>updated</i> road segments into the script (e.g. via osm2pgsql-replication update --changed-from=\$(last hex build)); recompute coverage for affected hexes only.
All-inside-PostGIS	If you later install the postgis_h3 extension, you can replace steps 2-4 with a single SQL SELECT h3.polyfill(geom,8), sum(length_km)... GROUP BY 1 and then COPY TO PROGRAM 'duckdb /parquet_writer ...'.

1.2.2 Export the file

If we had Parquet support in GDAL, the command below would have been smth like:

```
ogr2ogr -f GeoJSON cov.geojson "Parquet:cyprus_res8_roadkm.parquet"
```

Instead, we use direct export from Python for the time being:

```

[2]: import geopandas as gpd
gpd.read_parquet("cyprus_res8_roadkm.parquet").to_file("cov.geojson",
↳driver="GeoJSON") # streams via Fiona

```

1.3 Key take-aways

- **Stateless build step.** Both exports are pure reads—no schema changes, no temporary tables.
- **Stream-friendly.** ogr2ogr never loads the whole table/file into memory; good for future larger regions.

```
# completely in-memory pipeline (no temp files)
ogr2ogr -f GeoJSONSeq /vsistdout/ "PG:...cyprus..." -sql "SELECT ..." \
| tippecanoe -o cyprus_roads.pmtiles -l roads -
```

- **CRS consistency.** Everything leaves the data prep step in EPSG 4326, so Tippecanoe doesn't need extra flags.
- **Repeatability.** Can stick the two commands in a Makefile target (make geojson) and rerun whenever the PostGIS diff-tables or the Parquet metric update.

2 Build tiles (PMTiles)

2.1 Install tippecanoe

```
# 1. Refresh package lists
sudo apt update
# 2. Make sure the "universe" repo is enabled (it usually is, but this won't hurt)
sudo add-apt-repository universe
# 3. Grab the binary package
sudo apt install -y tippecanoe
# 4. Confirm it works
tippecanoe -v
# -> tippecanoe v2.53.0 (or whatever the current repo version is)
```

For Ubuntu 22.04 it's not available, so building from source:

```
sudo apt update \
&& sudo apt install -y build-essential git zlib1g-dev libsqlite3-dev

git clone https://github.com/mapbox/tippecanoe.git
cd tippecanoe
make PMTILES=1 -j"$(nproc)" # the PMTILES flag does not help for whatever reason
sudo make install
tippecanoe -v # → v1.36.0 which is functionally the same as 2.x Debian builds
tippecanoe --help | grep -E 'cluster-distance|detect-shared-borders|hilbert'
# If those flags are present, you're on the modern codebase no matter what the banner says.
# Something's fucked up with the --help output, me to check what iostream it uses later
# (venv) papavova@Thinker:/mnt/c/Users/papa.vova/tippecanoe$ tippecanoe --help > t.hlp 2>&1
# (venv) papavova@Thinker:/mnt/c/Users/papa.vova/tippecanoe$ cat t.hlp | \
# grep -E 'cluster-distance|detect-shared-borders|hilbert'
#      [--drop-polygons] [--cluster-distance=...]
#      [--detect-shared-borders] [--grid-low-zooms]
#      [--reverse] [--hilbert]
```

2.2 Install pmtiles

Go to <https://github.com/protomaps/go-pmtiles/releases/latest>, grab the linux binary.

Move it somewhere appropriate (ideally not on ntfs but ext4):

```
mv pmtiles ~
```

2.3 Export the tiles

```
# Zoom levels:
# * z6 shows the whole island;
# * z14 is the last level before "building footprints" detail
tippecanoe \
  -Z6 -z14 \
  --drop-densest-as-needed \
  --coalesce-densest-as-needed \
  -l roads \
  -o cyprus_roads.mbtiles \
  roads.geojson
```

Typical output:

```
(venv) papavova@Thinker:/mnt/c/Users/papa.vova$ tippecanoe \
Z6 -z14> -Z6 -z14 \
> --drop-densest-as-needed \
> --coalesce-densest-as-needed \
> -l roads \
> -o cyprus_roads.mbtiles \
> roads.geojson
153786 features, 9933493 bytes of geometry, 1561 bytes of separate metadata,
  223973 bytes of string pool
tile 6/37/25 size is 619837 with detail 12, >500000
Going to try keeping the sparsest 72.60% of the features to make it fit
tile 6/37/25 size is 537404 with detail 12, >500000
Going to try keeping the sparsest 60.79% of the features to make it fit
tile 7/75/50 size is 844528 with detail 12, >500000
Going to try keeping the sparsest 53.28% of the features to make it fit
tile 7/75/50 size is 553848 with detail 12, >500000
Going to try keeping the sparsest 43.29% of the features to make it fit
tile 8/151/101 size is 1090143 with detail 12, >500000
Going to try keeping the sparsest 41.28% of the features to make it fit
tile 8/151/101 size is 565914 with detail 12, >500000
Going to try keeping the sparsest 32.82% of the features to make it fit
tile 9/303/202 size is 629167 with detail 12, >500000
Going to try keeping the sparsest 71.52% of the features to make it fit
99.9% 14/9688/6500
(venv) papavova@Thinker:/mnt/c/Users/papa.vova$
```

Convert MBTiles to PMTiles (if direct save to PMTiles does not work)

```
(venv) papavova@Thinker:/mnt/c/Users/papa.vova$ ~/pmtiles convert cyprus_roads.mbtiles \
cyprus_roads.pmtiles
2025/06/03 11:24:05 convert.go:159: Pass 1: Assembling TileID set
2025/06/03 11:24:05 convert.go:190: Pass 2: writing tiles
100% | (3467/3467, 644 it/s)
2025/06/03 11:24:10 convert.go:244: # of addressed tiles: 3467
2025/06/03 11:24:10 convert.go:245: # of tile entries (after RLE): 3467
2025/06/03 11:24:10 convert.go:246: # of tile contents: 3467
2025/06/03 11:24:10 convert.go:269: Total dir bytes: 7228
2025/06/03 11:24:10 convert.go:270: Average bytes per addressed tile: 2.08
2025/06/03 11:24:11 convert.go:239: Finished in 5.586662787s
(venv) papavova@Thinker:/mnt/c/Users/papa.vova$
```

2.4 Checking the file

```
hexdump -C -n 8 cyprus_roads.<<<ext>>>
# 00000000 53 51 4c 69 74 65 20 66 + "SQLite f" + it's an MBTiles DB
# or
# 00000000 50 4d 54 30 ... + "PMT0" + already PMTiles
```

Specifically PMTiles:

```
(venv) papavova@Thinker:/mnt/c/Users/papa.vova$ ~/pmtiles show cyprus_roads.pmtiles
pmtiles spec version: 3
tile type: mvt
bounds: (long: 32.270037, lat: 34.563788) (long: 34.587549, lat: 35.695046)
min zoom: 6
max zoom: 14
center: (long: 33.365479, lat: 35.164827)
center zoom: 14
addressed tiles count: 3467
tile entries count: 3467
tile contents count: 3467
clustered: true
internal compression: gzip
tile compression: gzip
generator tippecanoe v1.36.0
maxzoom 14
format pbf
generator_options tippecanoe -Z6 -z14 --drop-densest-as-needed --coalesce-densest-as-needed
-l roads -o cyprus_roads.mbtiles roads.geojson
minzoom 6
name cyprus_roads.mbtiles
tilestats <object...>
type overlay
vector_layers <object...>
version 2
description cyprus_roads.mbtiles
(venv) papavova@Thinker:/mnt/c/Users/papa.vova$
```

3 Render with pmtiles+MapLibre

3.1 Serve tiles locally with pmtiles

```
# mkdir ~/tiles
# mv cyprus_roads.pmtiles ~/tiles/
cd ~/tiles
../pmtiles serve . --cors='*' \
    --public-url='http://localhost:8080'
```

pmtiles is a single static binary, so I can copy the exact file into e.g. a Cloud Run container later

Test the http://localhost:8080/cyprus_roads.json URL:

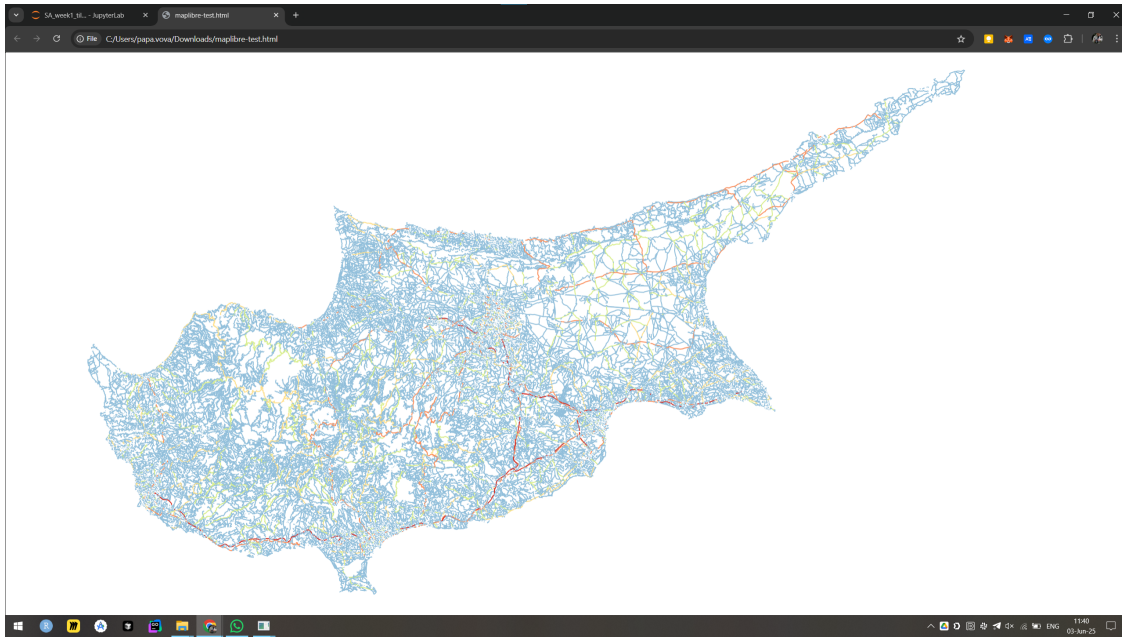
```
{
  "bounds": [
    32.270037,
    34.563788,
    34.587549,
    35.695046
  ],
  "center": [
    33.365479,
    35.164827,
    14
  ],
  "description": "cyprus_roads.mbtiles",
  "maxzoom": 14,
  "minzoom": 6,
  "name": "cyprus_roads.mbtiles",
  "scheme": "xyz",
  "tilejson": "3.0.0",
  "tiles": [
    "http://localhost:8080/cyprus_roads/{z}/{x}/{y}.mvt"
  ],
  "vector_layers": [
    {
      "description": "",
      "fields": {
        "highway": "String",
        "name": "String"
      },
      "id": "roads",
      "maxzoom": 14,
      "minzoom": 6
    }
  ],
  "version": "2"
}
```

3.2 Render

Minimal MapLibre page:

```
<!doctype html><html><head>
  <meta charset="utf-8">
  <link href="https://unpkg.com/maplibre-gl@3.5.2/dist/maplibre-gl.css" rel="stylesheet">
  <style>html,body,#map{height:100%;margin:0}</style>
</head><body><div id="map"></div>
<script src="https://unpkg.com/maplibre-gl@3.5.2/dist/maplibre-gl.js"></script>
<script>
  const map = new maplibregl.Map({
    container:"map", center:[33.3,35.15], zoom:7,
    style:{
      "version":8,
      "sources":{
        "roads":{
          "type":"vector",
          "url":"http://localhost:8080/cyprus_roads.json"    // <-- TileJSON!
        }
      },
      "layers":[{
        "id":"roads-line","type":"line","source":"roads","source-layer":"roads",
        "paint":{
          "line-color":["match",["get","highway"],
            "motorway","#d73027","primary","#fc8d59","secondary","#fee08b",
            "tertiary","#d9ef8b","#91bfdb"],
          "line-width":["interpolate",["linear"],["zoom"],6,0.4,14,4]
        }
      }]
    }
  });
</script></body></html>
```

Local copy: file:///C:/Users/papa.vova/Downloads/maplibre-test.html



4 Render-Alternative: direct PMTiles loading to MapLibre

4.1 (did not help, skip) Update Python to 3.12 to handle HTTP range requests

```
sudo apt-add-repository ppa:deadsnakes/ppa
sudo apt install python3.12
```

Then:

```
# 1 - Register both interpreters with priorities
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.10 110
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.12 120
# 2 - Pick the one you want as default
sudo update-alternatives --config python3
# choose the 3.12 line from the menu
# 3 - Verify
python3 --version      # → Python 3.12.x
which python3          # → /usr/bin/python3
```

!!! Keep system-wide Python3.10 !!!

Notabene: still needed `rangehttpserver` after all because of some open issue in Python server

```
pip install rangehttpserver      # installs a patched handler
```

4.2 Serve with Python / range-enabled server

```
cd ~/tiles
python -m RangeHTTPServer 8000
```

```
# Check, should show content-range:
papa.vova@Thinker:/mnt/c/Users/papa.vova$ curl -I -H "Range: bytes=0-15" \
```

```
http://localhost:8000/cyprus_roads.pmtiles
HTTP/1.0 206 Partial Content
Server: SimpleHTTP/0.6 Python/3.10.12
Date: Tue, 03 Jun 2025 09:55:18 GMT
Content-type: application/octet-stream
Content-Range: bytes 0-15/21500152
Content-Length: 16
Last-Modified: Tue, 03 Jun 2025 08:24:11 GMT
Accept-Ranges: bytes
```

4.3 Render

Create direct-range-loading.html:

```
<!DOCTYPE html><html><head>
<meta charset="utf-8"/>
<title>Cyprus roads - PMTiles + MapLibre demo</title>

<!-- MapLibre GL -->
<link href="https://unpkg.com/maplibre-gl@5.5.0/dist/maplibre-gl.css" rel="stylesheet"/>
<script src="https://unpkg.com/maplibre-gl@5.5.0/dist/maplibre-gl.js"></script>

<style>html,body,#map{height:100%;margin:0}</style>
</head><body><div id="map"></div>

<script type="module">
  /* 1. install the custom protocol */
  import { Protocol } from "https://unpkg.com/pmtiles@4.3.0/dist/pmtiles.mjs";
  const protocol = new Protocol();
  maplibregl.addProtocol("pmtiles", protocol.tile);

  /* 2. build the map */
  const map = new maplibregl.Map({
    container: "map",
    center: [33.3, 35.15],    // Cyprus
    zoom: 7,
    style: {
      "version": 8,
      "sources": {
        "roads": {
          "type": "vector",
          /* pmtiles:// tells the protocol to range-read the file */
          "url": "pmtiles://cyprus_roads.pmtiles"
        }
      },
      "layers": [{
        "id": "roads-line",
        "type": "line",
```

Concern	Talking point you can raise in the interview
---------	--

Hosting	<i>Cheapest:</i> store <code>.pmtiles</code> in a public GCS bucket and let MapLibre read it directly via HTTP range requests; no server needed. <i>Better:</i> run <code>go-pmtiles</code> on Cloud Run so you can add Cloud Front-style caching headers and metrics.
Basemap vs overlay	Keep Cyprus roads as a lightweight overlay on top of a Protomaps basemap; same protocol, zero CORS pain.
File size vs MBTiles	PMTiles clusters indexes, so you'll usually see 20–40 % smaller archives and 1-RTT tile reads; bring the actual <code>ls -lh</code> numbers.
Fallback to raster	MapLibre style can switch the source from vector to raster in one line (<code>"type": "raster"</code>) if older devices choke on WebGL.
Why Tippecanoe	Actively maintained, supports attribute-driven generalisation, and — crucial for PMTiles — directly writes the format since v2.17 .
