GeoSpatial Set Up

Contents

[Output Expected 2](#_Toc210459962)

[Docker Set up 2](#_Toc210459963)

[Pre-requisites 2](#_Toc210459964)

[Global Map (Perform this step only when a full world map is needed and Docker can allocate enough storage) 6](#_Toc210459965)

[React Set Up in Visual Studio Code 7](#_Toc210459966)

# Output Expected

A map of the north and south america

AI-generated content may be incorrect.

Docker Link:

# Docker Set up

## Pre-requisites

Before running any of the steps, make sure you have these ready:

1. **Docker Desktop installed on Windows**
   * Download: https://www.docker.com/products/docker-desktop
   * During installation, enable **WSL 2 backend**.
2. **WSL2 Linux distribution installed**
   * Check your distro (e.g., Ubuntu) with:
   * wsl -l -v
   * If it’s not version 2, upgrade:
   * wsl --set-version Ubuntu-20.04 2
3. **Enable WSL integration in Docker Desktop**
   * Open Docker Desktop → Settings → **Resources → WSL Integration**
   * Enable your distro (e.g., Ubuntu)
   * Restart Docker Desktop.
4. **Test Docker in WSL**  
   Inside your WSL terminal:
5. docker --version
6. docker run hello-world

If “Hello from Docker!” appears, you’re good to go.

1. **Other essentials**
   * **Node.js LTS** for the React frontend (node -v, npm -v)
   * **VS Code** with WSL extension (for easier editing)
   * **wget / curl** installed in your WSL (to fetch data)
   * **Enough disk space**: ~10GB for regional extracts, ~100GB+ if building full planet.
   * **RAM/CPU**: at least 8GB RAM, 4 cores (world builds need more: 32GB+ RAM ideally).

**1. Create project folders (WSL)**

mkdir -p ~/geo-map-starter/data

cd ~/geo-map-starter

**2. Download OSM extract (example: Singapore/Malaysia/Brunei)**

wget https://download.geofabrik.de/asia/malaysia-singapore-brunei-latest.osm.pbf -O data/region.osm.pbf

👉 Result: data/region.osm.pbf

**3. Build MBTiles with Planetiler (OpenMapTiles schema)**

docker run --rm -it \

-e JAVA\_TOOL\_OPTIONS="-Xmx4g" \

-v "$(pwd)/data":/data \

ghcr.io/onthegomap/planetiler:latest \

--osm-path=/data/region.osm.pbf \

--download \

--output=/data/openmaptiles.mbtiles \

--force

👉 Result: data/openmaptiles.mbtiles

**4. Create config.json**

nano ~/geo-map-starter/data/config.json

Paste this:

{

"options": {

"paths": {

"root": "/usr/src/app/node\_modules/tileserver-gl-styles",

"fonts": "fonts",

"styles": "styles",

"mbtiles": "/data"

}

},

"styles": {

"dark-matter": {

"style": "/data/styles/dark-matter/style.json"

}

},

"data": {

"openmaptiles": {

"mbtiles": "/data/openmaptiles.mbtiles"

}

}

}

👉 Lessons fixed:

* Removed "sprites" (caused missing path error).
* Used absolute path /data/styles/dark-matter/style.json.

**5. Add Dark Matter style.json**

Make the folder:

mkdir -p ~/geo-map-starter/data/styles/dark-matter

nano ~/geo-map-starter/data/styles/dark-matter/style.json

Paste the **verified JSON**:

{

"version": 8,

"name": "Dark Matter",

"metadata": {

"maputnik:renderer": "mbgljs"

},

"sources": {

"openmaptiles": {

"type": "vector",

"url": "mbtiles://{openmaptiles}"

}

},

"glyphs": "http://localhost:8080/fonts/{fontstack}/{range}.pbf",

"sprite": "https://demotiles.maplibre.org/styles/osm-bright-gl-style/sprite",

"layers": [

{

"id": "background",

"type": "background",

"paint": {

"background-color": "#000000"

}

},

{

"id": "land",

"type": "fill",

"source": "openmaptiles",

"source-layer": "land",

"paint": {

"fill-color": "#111111"

}

},

{

"id": "water",

"type": "fill",

"source": "openmaptiles",

"source-layer": "water",

"paint": {

"fill-color": "#222222"

}

},

{

"id": "roads",

"type": "line",

"source": "openmaptiles",

"source-layer": "transportation",

"paint": {

"line-color": "#444444",

"line-width": 1

}

},

{

"id": "boundaries",

"type": "line",

"source": "openmaptiles",

"source-layer": "boundary",

"paint": {

"line-color": "#666666",

"line-width": 0.5

}

},

{

"id": "places",

"type": "symbol",

"source": "openmaptiles",

"source-layer": "place",

"layout": {

"text-field": "{name}",

"text-size": 12,

"text-font": ["Noto Sans Regular"]

},

"paint": {

"text-color": "#cccccc"

}

}

]

}

Save → Ctrl+O → Enter → Ctrl+X.

👉 Lessons fixed:

* Without this file, styles return **Not Found**.
* This JSON matches OpenMapTiles schema.

5b. Add sprites for icons **(Optional — no need to do this now**. Later, the glyphs and sprite entries in *style.json* can be updated to replace the current localhost references.)

# 1) Make sprites folder

mkdir -p ~/geo-map-starter/data/styles/dark-matter/sprites

# 2) Download sprite files

* https://demotiles.maplibre.org/styles/osm-bright-gl-style/sprite.json
* https://demotiles.maplibre.org/styles/osm-bright-gl-style/sprite.png
* https://demotiles.maplibre.org/styles/osm-bright-gl-style/sprite@2x.json
* https://demotiles.maplibre.org/styles/osm-bright-gl-style/sprite@2x.png

cd ~/geo-map-starter/data/styles/dark-matter/sprites

wget -q https://demotiles.maplibre.org/styles/osm-bright-gl-style/sprite.json -O sprite.json

wget -q https://demotiles.maplibre.org/styles/osm-bright-gl-style/sprite.png -O sprite.png

wget -q https://demotiles.maplibre.org/styles/osm-bright-gl-style/sprite@2x.json -O sprite@2x.json

wget -q https://demotiles.maplibre.org/styles/osm-bright-gl-style/sprite@2x.png -O sprite@2x.png

**6. Run TileServer-GL with config**

cd ~/geo-map-starter

docker stop tileserver 2>/dev/null || true

docker run --name tileserver \

--rm -it -p 8080:8080 \

-v "$(pwd)/data":/data \

maptiler/tileserver-gl:latest \

-c /data/config.json

👉 Logs should include:

Using specified config file from /data/config.json

Startup complete

**7. Verify**

* [http://localhost:8080](http://localhost:8080/) → Dashboard lists **dark-matter**
* <http://localhost:8080/styles/dark-matter/style.json> → JSON displays

Tile server + Dark Matter style working.

**Summary of Lessons Learned**

1. **Sprites error** → fixed by removing "sprites" path.
2. **Style missing** → fixed by creating our own style.json.
3. **Path error** → fixed by using absolute /data/styles/... path in config.
4. **Config ignored** → fixed by running with -c /data/config.json.
5. **Now final result**: OSM tiles + Dark Matter style load correctly in TileServer-GL.

## Global Map (Perform this step only when a full world map is needed and Docker can allocate enough storage)

**Replace Step 2 — Download the Global OSM Extract (Planet)**

# from ~/geo-map-starter

wget https://planet.openstreetmap.org/pbf/planet-latest.osm.pbf -O data/planet.osm.pbf

# Result: data/planet.osm.pbf (very large file)

**Replace Step 3 — Build MBTiles with Planetiler (Global/World)**

The command is the same as your regional build, but point to the planet file and give Planetiler more memory.

docker run --rm -it \

-e JAVA\_TOOL\_OPTIONS="-Xmx16g" \

-v "$(pwd)/data":/data \

ghcr.io/onthegomap/planetiler:latest \

--osm-path=/data/planet.osm.pbf \

--download \

--output=/data/openmaptiles.mbtiles \

--force

# Result: data/openmaptiles.mbtiles (global tiles)

**Additional Notes for “Global / World” Builds**

* **Hardware/Time**: World builds are heavy. Expect many hours, lots of CPU, fast SSD, and *preferably* 32–64 GB RAM (the -Xmx16g above is a practical minimum; more is better).
* **WSL**: If you’re on WSL2, ensure your .wslconfig allows enough RAM/CPU; otherwise the build can fail or thrash.
* **No other changes required**: Your config.json, dark-matter style.json, and TileServer-GL run command stay the same. After the build, just (re)start:
* **Attribution**: Keep © OpenStreetMap contributors visible in your MapLibre app (you already have this).

# React Set Up in Visual Studio Code

**Prerequisites**

* Docker Desktop installed and running
* Node.js LTS installed (node -v, npm -v should work)
* Visual Studio Code installed

**1. Start TileServer-GL**

In a terminal (WSL/Linux/macOS):

cd ~/geo-map-starter

docker run --name tileserver \

--rm -it -p 8080:8080 \

-v "$(pwd)/data":/data \

maptiler/tileserver-gl:latest \

-c /data/config.json

Check:

* [http://localhost:8080](http://localhost:8080/) → dashboard
* <http://localhost:8080/styles/dark-matter/style.json> → style JSON loads

Keep this container running.

**2. Create React project**

Open **VS Code**, then:

cd ~/geo-map-starter

npx create-react-app react-map

cd react-map

npm install maplibre-gl

**3. Add MapLibre CSS**

Edit src/index.js:

import 'maplibre-gl/dist/maplibre-gl.css';

**4. Add Map component**

Create src/Map.js:

import { useEffect, useRef } from "react";

import maplibregl from "maplibre-gl";

export default function Map() {

const mapContainer = useRef(null);

useEffect(() => {

if (!mapContainer.current) return;

const map = new maplibregl.Map({

container: mapContainer.current,

style: "http://localhost:8080/styles/dark-matter/style.json",

center: [103.851959, 1.290270], // Singapore

zoom: 10

});

map.addControl(new maplibregl.NavigationControl(), "top-right");

map.addControl(

new maplibregl.AttributionControl({

compact: true,

customAttribution: "© OpenStreetMap contributors"

}),

"bottom-right"

);

return () => map.remove();

}, []);

return <div ref={mapContainer} style={{ width: "100vw", height: "100vh" }} />;

}

**5. Wire Map into App**

Replace src/App.js:

import Map from "./Map";

function App() {

return <Map />;

}

export default App;

**6. Run the app**

From VS Code terminal:

cd react-map

npm start

Open [http://localhost:3000](http://localhost:3000/) → **Dark Matter map** renders.

**Notes**

* Ensure the TileServer container is running on port 8080.
* Change ports if necessary (-p 8090:8080 → then update the style URL).
* OSM attribution must be visible (© OpenStreetMap contributors).
* For production: npm run build → deploy build/ folder.