

MapSplat User Guide

MapSplat — User Guide

MapSplat turns the layers in your QGIS project into a self-contained web map — a folder of static files (vector tiles in **PMTiles** format + an **HTML viewer** built on **MapLibre GL JS**) that you can open in any browser or drop on any static web host. No tile server, no backend.

1. What you need

Requirement	Why	Notes
QGIS 4.0+	The plugin runs inside QGIS	PyQt6 build
GDAL 3.8+	Converts your vector layers to PMTiles	Ships with QGIS; nothing to install
<code>pmtiles</code> CLI	Only for the optional <i>Basemap Overlay</i>	Install from the releases page and put it on your <code>PATH</code> . Core export does not need it.

You do **not** need Node, a database, or any web stack.

2. Quick start (2 required steps)

Open **Plugins ► MapSplat** (or the toolbar button) to show the dock, then on the **Inputs** tab:

1. ① **Layers** — tick the vector layers you want to publish. Their QGIS **styles and labels are read automatically**. (When you open the dock, your currently-visible layers are pre-selected.)
2. ② **Output** — set a **Project name** and an **Output folder**. The export is written to `<output folder>/<project name>_webmap/`. (These are pre-filled from your project — adjust if needed.)

The **readiness line** above the Export button tells you what's still missing; when it turns green (“*Ready to export*”) the button enables. Click **Export Web Map**, watch the **Log** tab, then **Open Folder** to see the result.

That's the whole happy path. Everything below is optional and has sensible defaults.

3. The dock, tab by tab

- **Inputs** — Layers, Output, and the Export button. A full run lives here.
 - **Options** — *Export Options* (PMTiles mode, max zoom, tile-count estimate, style.json, export extent) and *Basemap Overlay*. Defaults are fine for most maps.
 - **Viewer** — what the generated web map shows: scale bar, geolocate, fullscreen, coordinate/zoom readouts, reset/north buttons, label placement, legend, attribution, and map dimensions.
 - **Offline** — bundle MapLibre/PMTiles JS + CSS into the export so the viewer works with no internet.
 - **Log** — progress, messages, and the **version stamp** (bottom-right) so you can confirm which build is loaded.
-

4. Key options explained

- **PMTiles mode** — *Single file* merges all layers into one `.pmtiles`; *Separate files* writes one per layer (loaded independently in the viewer).
 - **Max zoom** — higher = more detail but **exponentially** more tiles/time. 6–10 suits most maps; 14+ can take a long time on large data. The live estimate under it shows the rough tile count/size.
 - **Export extent** — clip the basemap to a chosen layer's extent or the current map view instead of the full data extent.
 - **Style only** (*Advanced*) — regenerate `style.json` + the viewer without re-tiling the data.
-

5. Adding a Protomaps basemap (optional)

A basemap gives your data context (streets, water, terrain). MapSplat uses **Protomaps** — free, OpenStreetMap-derived vector basemaps in PMTiles format.

Where to get the tiles. Protomaps publishes daily global builds and an area extractor at build.protomaps.com:

- *Small area (recommended)* — use the map on that page to draw your region and download a **trimmed .pmtiles** for just that area. Small and fast.
- *Whole planet* — download the full dated build (e.g. `20260401.pmtiles`). Large; MapSplat will clip it.

Point MapSplat at either a **local file** you downloaded or a **remote build URL**.

Where to get the style. You also need a Protomaps-compatible MapLibre `style.json` (colours, fonts, which basemap layers to draw). Protomaps ships ready-made styles (light, dark, etc.) — see docs.protomaps.com/basemaps.

How MapSplat uses them. On **Options ▶ Basemap Overlay**, enable it and set the **source** and **basemap style.json**. At export, MapSplat runs `pmtiles extract --bbox` to clip the basemap to your

data's bounding box — so the offline map carries only the tiles it needs — then overlays your layers on top.

This step needs the `pmtiles` CLI. The clip shells out to the `pmtiles` command. MapSplat no longer *bundles* that program (QGIS forbids shipping executables in plugins), so install it once from the [go-pmtiles releases](#) and put it on your `PATH`. Exporting *your* layers uses GDAL and needs **no** CLI — only the basemap overlay does. If you'd rather not install it, just publish your data without a basemap.

6. Viewing / serving the output

The export folder contains `index.html`, a `data/` folder of `.pmtiles`, and (optionally) a `lib/` folder of bundled JS/CSS. Because PMTiles uses HTTP **Range** requests, opening `index.html` directly from disk may not work — serve it over HTTP:

- **Bundled dev server:** run `serve.py` in the export folder (`python serve.py`) and open the printed URL.
 - **Any static host** that supports range requests: Netlify, Cloudflare Pages, S3/CloudFront, GitHub Pages, nginx, etc. Just upload the folder.
-

7. Troubleshooting

- **“pmtiles CLI not found”** — only needed for the basemap. Install it and ensure QGIS sees your `PATH`. If it works in a terminal but not from the QGIS *Python Console* (`shutil.which("pmtiles")` returns `None`), launch QGIS from a terminal so it inherits your full shell `PATH`.
 - **Export is huge / slow** — lower **Max zoom**; watch the tile estimate.
 - **Blank map in the browser** — you opened `index.html` from disk; serve it over HTTP (§6).
 - **The dock looks unchanged after an update** — QGIS caches plugin code; **fully restart QGIS** (or use *Plugin Reloader*). Confirm via the **version stamp** on the Log tab.
 - **A layer's symbology didn't translate** — a  icon in the layer list flags renderers/markers (heatmap, point cluster, font markers...) that don't map cleanly to MapLibre.
-

8. Styling — what carries over, and what doesn't

MapSplat reads each layer's QGIS symbology and labels and converts them to a MapLibre style. Most everyday styling translates well; a few QGIS features have no MapLibre equivalent. Layers with symbology that won't translate cleanly are flagged with a  icon in the layer list (hover for why).

Translates well - Single-symbol, categorized, graduated, and rule-based renderers. - Fill and stroke colours, widths, opacity, and simple line/dash patterns. - **SVG markers** — converted to a sprite sheet — and basic marker/line/fill symbols. - Labels: text, font, size, colour, and halo.

Limited or not supported - **Heatmap renderer** — exported as circle markers, not a smooth heatmap. - **Point cluster renderer** — clustering isn't reproduced; points render at their true positions. - **Point displacement renderer** — displaced positions aren't preserved. - **Font markers** — render as a plain circle (use an SVG marker for a custom glyph). - **Draw effects** (drop shadow, glow, blur), **blend modes**, the **2.5D** renderer, and **geometry generators** — no MapLibre equivalent; ignored. - **Data-defined (expression) overrides** on symbol properties — only simple cases translate. - Very complex or deeply nested rule sets may be simplified.

Tip: for the most faithful web map, favour categorized / graduated / rule renderers with solid fills, strokes, and SVG markers, and keep data-defined symbology simple.

9. Versions this build was made with

Component	Version
MapSplat	0.19.0
MapLibre GL JS (viewer)	5.24.0
PMTiles JS (viewer)	4.4.1
pmtiles CLI (basemap only; tested)	1.30.1
QGIS	4.0+ required — built and tested on 4.2
GDAL	3.8+ required (PMTiles driver) — tested on 3.12
Qt / Python bindings	PyQt6 (QGIS 4)

The MapLibre and PMTiles **JS** versions are pinned in the generated viewer; the `pmtiles` **CLI** is whatever you have installed on your `PATH`.

MapSplat is free software (GPL-2.0-or-later). Source, issues, and updates: <https://github.com/johnzastrow/mapsplat4>.