Engineer

GDAL

a library for reading and converting geospatial data

What it is

GDAL, short for Geospatial Data Abstraction Library and often written GDAL/OGR, is open-source software that opens map data stored in one format and writes it out in another, reprojecting it on the way when asked. Its own documentation calls it a translator library.

Spatial data comes in two shapes. Raster data is a grid of cells: a satellite image, an aerial photo, an elevation model. Vector data is points, lines and polygons with attributes attached: addresses, roads, parcel boundaries. GDAL gives the program that calls it one abstract model for each shape, and behind that model sits a driver per format. GeoTIFF, JPEG 2000 and NetCDF are drivers on the raster side; Shapefile, GeoJSON, GeoPackage, KML and PostgreSQL tables are drivers on the vector side, and there are many more of both. A program written against GDAL reads all of them through the same calls.

The vector half was known for years as OGR, and the two halves kept separate driver structures until GDAL 2.0 merged them. That history is why the vector programs still carry the old prefix: `ogrinfo` describes a vector dataset and `ogr2ogr` converts one. Work on the library began in late 1998 with Frank Warmerdam. It is released under an MIT-style licence and is a project of the Open Source Geospatial Foundation, OSGeo.

There are three ways to use it. The first is the command line, through programs such as `gdalinfo`, `gdal_translate` and `gdalwarp` for rasters and `ogrinfo` and `ogr2ogr` for vectors. Since version 3.11 there is also a single `gdal` command with subcommands such as `gdal raster` and `gdal vector`, which the project still marks as provisional. The second is code, through bindings for Python and other languages. The third is inside other software: QGIS, for example, reads its vector

formats through the GDAL drivers and offers the GDAL programs in its Processing toolbox. A typical conversion fits on one line, here turning a shapefile into a GeoPackage and reprojecting it to the UTM zone used across Denmark:

```
ogr2ogr -f GPKG -t_srs EPSG:25832 parcels.gpkg parcels.shp
```

At the time of writing the current release is 3.13.3. Few tools compete at this layer. ArcGIS ships its own conversion tools and single formats have small readers of their own, but a team that must accept whatever a client sends tends to end up with GDAL in its stack, often through a library that depends on it.

Use the concept in a decision

Put “What is GDAL?” to work in a real decision. We can help relate the concept to the system you are building or buying. Anything that happens in a place — vessels, vehicles, plots, sensors — put on a map you can search, measure and share.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Made a geographical data pipeline 50x faster — 50x faster geographical data pipelines

Made a geographical data pipeline 50x faster Get in touch →

Why it matters

Every geospatial system has a border where outside data comes in: a municipality’s shapefiles, a surveyor’s GeoTIFFs, a partner’s GeoJSON export, a KML file someone drew in a browser. The decision GDAL bears on is who stands at that border. It can be a person clicking through a desktop tool, or a script that does the same thing every time. GDAL is what makes the script cheap to write.

A conversion written as an `ogr2ogr` or `gdalwarp` command can be read in review, kept in version control and run again when the next delivery arrives. A conversion done by hand is done once, by whoever remembers the steps. For data that arrives in batches, month after month, that difference decides whether quality holds.

The licence matters too. There is no per-seat fee and no vendor approval, and the same tool runs on a laptop, a build server and inside a container.

The third reason is reach into storage. GDAL's virtual file systems let a driver read data where it lies. With the `/vsicurl/` prefix it reads a file over HTTP by asking for byte ranges, so most drivers consume a remote dataset piece by piece without downloading all of it, and `/vsis3/` does the same against Amazon S3. Pair that with a Cloud Optimized GeoTIFF, which is tiled, carries its own overviews and is laid out to keep the number of requests low, and a raster of many gigabytes can sit in object storage and still be read one window at a time.

The costs are real as well. GDAL is a large native library with many optional dependencies, and which drivers a given build contains depends on how it was compiled, so two machines that both "have GDAL" can open different formats. And it translates faithfully, which includes faithfully carrying over every limit of the format it is asked to write.

What it looks like in practice

GDAL appears on two projects on this site, in two different roles.

From 2015 to 2022, as GIS Software Development Team Lead, delivery ran on more than thirty GIS projects, each one supported in production after it shipped. Underneath them sat PostgreSQL with PostGIS for the spatial data, and GDAL/OGR was the tool for moving that data between formats: shapefiles, GeoJSON, GeoTIFF, KML and vector tiles. QGIS and JOSM handled the data work itself, and the maps on top were built with Mapbox GL and Leaflet. Both halves of the library were in use, since GeoTIFF is raster and the rest of that list is vector, and the projects themselves ranged from 2D and 3D digital mapping and LiDAR processing to georectification, vectorisation, indoor mapping and navigation.

From 2022 to 2024, in a Data Engineer role on a large-scale geospatial analytics project, GDAL was one of the tools for finding and fixing errors. Spatial errors were diagnosed and corrected with GDAL, QGIS and ArcGIS, and because the same cleaning tasks kept coming back, they were automated as Bash scripts. PostGIS carried the spatial queries and indexes, procedures in PL/pgSQL and Transact-SQL transformed the data inside PostgreSQL and SQL Server, and the cleaned result fed interactive visualisations built with Mapbox. With GDAL in the toolkit, the cleanup resolved more than 1,000 issues, and it was one of the reasons spatial query run times came down by 35%.

Where it goes wrong

GDAL does exactly what it is told. Most trouble starts when it is told something slightly different from what was meant.

Points land on the wrong continent. Since GDAL 3.0 the library honours the axis order set by the authority that defines a coordinate system, and for EPSG:4326, the familiar latitude and longitude system, that order is latitude first. Code written for GDAL 2 passes longitude first, and so does GeoJSON. Mix the two and every pair is swapped: a point in Copenhagen reappears in the Arabian Sea. Decide on one order at the edge of the program and state it. In code that wants longitude first, the documented switch is `SetAxisMappingStrategy(OAMS_TRADITIONAL_GIS_ORDER)` on the spatial reference.

Column names come out cut short and dates lose their time. A shapefile keeps attribute names to 10 characters. GDAL shortens longer names as it writes, and when two shortened names collide it cuts them to 8 characters and adds a serial number, so `abcdefghijkl` becomes `abcdefgh_1`. The format stores dates without times. GDAL recommends keeping `.shp` and `.dbf` files at 2 GB or less for interoperability with other software. Danish or Cyrillic text can turn to noise if its encoding is misidentified when the `.cpg` file that names the code page is missing. Write GeoPackage where the receiver allows it; where a shapefile is required, rename the columns on purpose before converting.

A remote raster is downloaded whole. Given a plain `https://` address with no prefix, GDAL picks its HTTP pseudo-driver, which pulls the entire file into memory before any driver reads it. That suits a small file and is the wrong choice for a national orthophoto. Prefix the address with `/vsicurl/` so the driver fetches only the byte ranges it needs, and store large rasters as Cloud Optimized GeoTIFF so those ranges are few. Then leave the file alone: editing a COG in place breaks its layout, and since GDAL 3.8 an attempt to open one for update is refused unless that break is explicitly allowed.

A CSV’s numbers arrive as text. The CSV driver reads every field as a string unless told otherwise, so a column of populations sorts alphabetically and a sum downstream turns into a type error. Set the `AUTODETECT_TYPE=YES` open option, or ship a `.csvt` file beside the CSV that names each column’s type.

An upgrade breaks the Python import. The `gdal` package on PyPI is built against the GDAL already installed on the machine, so the two versions have to agree. The project's install instructions pin one to the other:

```
pip install gdal[numpy]=="$(gdal-config --version).*"
```

Further reading

- GDAL documentation — the project's own description of the translator library, its drivers and its licence.
- Programs — every command-line tool, from `gdalinfo` and `ogr2ogr` to the newer `gdal` command.
- GDAL Virtual File Systems — `/vsicurl/`, `/vsis3/` and the other prefixes for reading data where it lies.
- OGR Coordinate Reference Systems and Coordinate Transformation tutorial — with the section on axis order and what changed in GDAL 3.0.
- COG – Cloud Optimized GeoTIFF generator — the file layout, its overviews and why editing in place undoes them.

Glossary

Choose for your actual constraints

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is GDAL?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/gdal/