
GDAL

GDAL CLI Modernization

Recording of the webinar: https://www.youtube.com/watch?v=ZKdrYm3TiBU
("GDAL CLI Modernization" on OSGeo channel)

Even Rouault
Spatialys

Howard Butler
Hobu Inc.

Daniel Baston
ISciences, LLC

- Gold level:

planet.

esri

Microsoft

- Silver level:

Google

Safe Software

- Bronze level:

- Supporter level:

Dynamic Graphics, Inc. Umbra Vortex f.d.c. Route4Me, Inc. PIX4D Satelligence Space Intelligence Myles Sutherland T-Kartor Kaplan Open Source Consulting

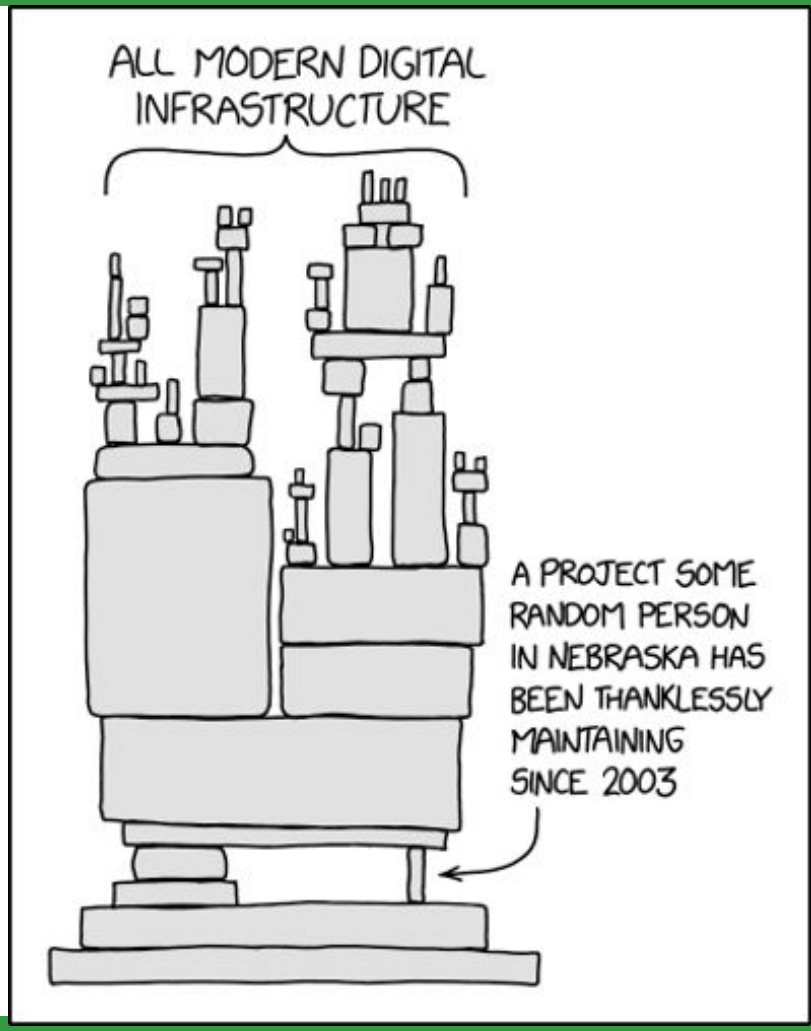
Regrid Phoenix LiDAR Systems, LLC

GDAL

Thank you sponsors!

GDAL Sponsorship Program

- Maintenance!
  - ~27 years of technical debt
  - Refactoring is hard
  - Behavior preservation
- No single organization will fund
  - Security remediation
  - Continuous integration
  - Documentation
  - *CLI refactor*

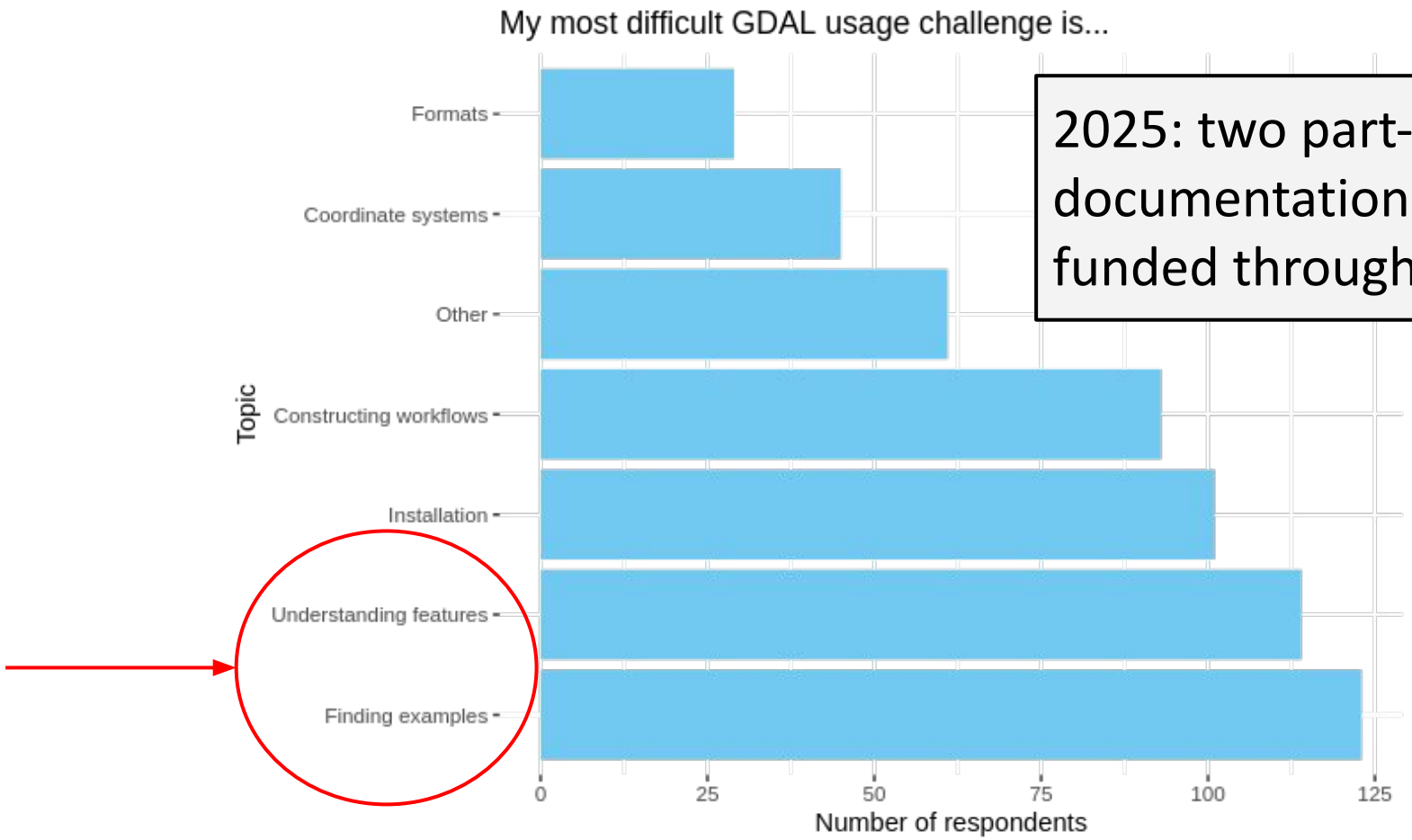


2024 GDAL user survey

- 600 respondents representing industry, government, academia
- Invaluable information about
  - Where do people obtain GDAL?
  - What platforms?
  - What tools do they use?
  - What would make GDAL better?
- Stay tuned for 2025 edition

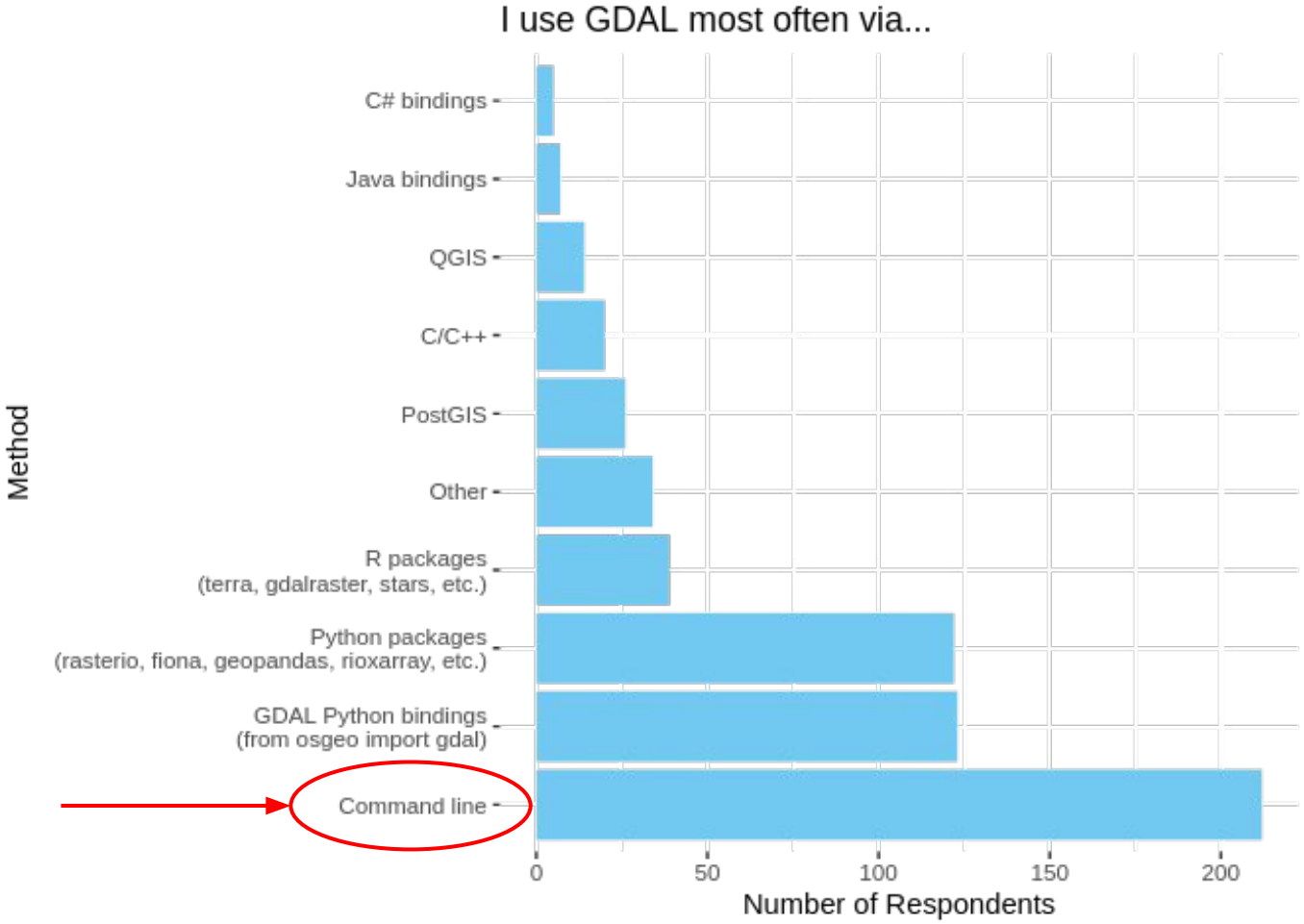
Summarized results available on the project website

GDAL needs more documentation!



2025: two part-time documentation writers funded through GSP!

CLI is the most popular way to use GDAL



Everybody ❤️ GDAL utilities!

- Powerful and comprehensive
  - Format conversion, reprojection, SQL querying, resampling, mosaicing, …
- Gracefully handle very large datasets
- 20+ years of backwards compatibility
- Library-callable from C/C++/Python

What could make GDAL easier to use?

A consistent interface, i.e. same name for same options across different commands, would help me.

gdal command -consistent --options

OMG the command line options on the tools. ***Burn the village to the ground and build again****.*

Everybody 🤬 GDAL utilities ! (1/4)

- **Underscore or not ?**
  - `gdal_translate` vs `gdalwarp`
  - `gdal_merge` vs `ogrmerge`
- **Dataset order inconsistency:**
  - `gdal_translate in.tif out.tif`
  - `gdalwarp out.tif in.tif`
- **Overwrite existing output file or not ?**
  - `gdal_translate`: Yes (silently)
  - `gdalwarp`: No, update (silently), unless `-overwrite`

Everybody 😡 GDAL utilities ! (2/4)

- **Inconsistent argument naming**:
  - `gdal_translate -projwin ulx uly lrx lry` **vs**
    `gdalwarp -te llx lly urx ury`
  - `gdal_translate -outsize width height` **vs**
    `gdalwarp -ts width height`
- **Too many things at once: how do I override the extent of a raster ?**
  - `gdal_translate -projwin ulx uly lrx lry` **or**
    `gdal_translate -a_ullr ulx uly lrx lry`

Everybody 🤬 GDAL utilities ! (3/4)

- 76! vector operations go through `ogr2ogr`
- Many mutually exclusive options
- Unpredictable order of chained operations

```
[-clipsrc [<xmin> <ymin> <xmax>
<ymax>]|<WKT>|<datasource>|spat_extent]
[-clipsrcsql <sql_statement>] [-clipsrclayer <layername>]
[-clipsrcwhere <expression>]

[-clipdst [<xmin> <ymin> <xmax> <ymax>]|<WKT>|<datasource>]
[-clipdstsql <sql_statement>] [-clipdstlayer <layername>]
-clipdstwhere <expression>]
```

Everybody 😡 GDAL utilities ! (4/4)

- Inconsistent arguments between C++ and Python utilities:
  - C++ utilities: `-single_dash`
  - Python utilities: `--double-dash`
- No completion or typo suggestions

RFC 104: Adding a `gdal` CLI

- Burn the village to the ground 🔥
- `git` -style sub-commands
- Break apart huge CLIs
  - `gdal_translate` and `ogr2ogr`
  - `gdalwarp` and `gdaldem`
- Unified CLI framework
- Programmatic discoverability
- GDALG pipelining

`gdal`: what's new? (1 / 3)

- Short options
  - single dash: `-i my.tif`
- Long options
  - double dash: `--update`
- Two possibilities to pass values
  - `--input=my.tif`
  - `--input my.tif`

`gdal`: what's new? (2 / 3)

- **Predictable input/output orders!**
  - **Input(s) first**
  - **Output last**

```
$ gdal raster convert in.gpkg out.tif
$ gdal vector convert in.shp out.gpkg
$ gdal raster convert -i in.gpkg -o out.tif
```

- **Overwrite always requires `--overwrite`**

`gdal`: what's new? (3 / 3)

- Always use hyphen (no more _'s)
- Detection of typos such as letter inversion

```
$ gdal raster convret
ERROR 1: Algorithm 'convret' is unknown. Do you
mean 'convert'?
```

```
$ gdal raster convert --creattionoption
ERROR 5: convert: Option '--creattionoption' is
unknown. Do you mean '--creation-option'?
```

First level commands

```
$ gdal

ERROR 1: gdal: Missing command name.
Usage: gdal <COMMAND> [OPTIONS]
where <COMMAND> is one of:
  - convert:  Convert a dataset (shortcut for 'gdal raster convert' or 'gdal
vector convert').
  - driver:   Command for driver specific operations.
  - info:     Return information on a dataset (shortcut for 'gdal raster info'
or 'gdal vector info').
  - mdim:     Multidimensional commands.
  - pipeline: Execute a pipeline (shortcut for 'gdal raster pipeline' or 'gdal
vector pipeline').
  - raster:   Raster commands.
  - vector:   Vector commands.
  - vsi:      GDAL Virtual System Interface (VSI) commands.

Try 'gdal --help' for help.

'gdal <FILENAME>' can also be used as a shortcut for 'gdal info <FILENAME>'.
And 'gdal read <FILENAME> ! ...' as a shortcut for 'gdal pipeline <FILENAME> !
...'.

For more details, consult https://gdal.org/programs/index.html
```

gdal raster (1 / 2)

```
- aspect:        Generate an aspect map  (gdaldem aspect)
- calc:          Perform raster algebra
- clean-collar:  Clean the collar, removing noise. (nearblack)
- clip:          Clip a raster dataset   (gdal_translate -projwin)
- color-map:     Generate a RGB or RGBA dataset from a single band, using
                 a color map (gdaldem color-relief or gdal_translate -expand rgb)
- contour:       Creates a vector contour from a raster elevation model (DEM)
- convert:       Convert a raster dataset (gdal_translate)
- create:        Create a new raster dataset (gdal_create)
- edit:          Edit a raster dataset (gdal_translate -mo/-a_nodata/-a_srs)
- fill-nodata:   Fill nodata raster regions by interpolation from edges.
                 (gdal_fillnodata.py)
- footprint:     Compute the footprint of a raster dataset (gdal_footprint)
- hillshade:     Generate a shaded relief map (gdaldem hillshade)
- index:         Create a vector index of raster datasets (gdaltindex)
- info:          Return information on a raster dataset (gdalinfo)
- mosaic:        Build a mosaic, virtual (VRT) or materialized. (gdalbuildvrt)
- overview:      Manage overviews of a raster dataset (gdaladdo)
- pipeline:      Process a raster dataset.
- pixel-info:    Return information on a pixel (gdallocationinfo)
- polygonize:    Create a polygon feature dataset from a band (gdal_polygonize)
- reclassify:    Reclassify values in a raster dataset
- reproject:     Reproject a raster dataset (gdalwarp)
```

gdal raster (2 / 2)

```
- resize:        Resize a raster dataset without changing the
                 georeferenced extents  (gdal_translate -outsize)
- roughness:     Generate a roughness map  (gdaldem roughness)
- scale:         Scale the values of the bands  (gdal_translate -scale)
- select:        Select a subset of bands  (gdal_translate -b)
- set-type:      Modify the data type of bands  (gdal_translate -ot)
- sieve:         Remove small polygons  (gdal_sieve.py)
- slope:         Generate a slope map  (gdaldem slope)
- stack:         Combine together input bands into a multi-band output,
                 virtual (VRT) or materialized  (gdalbuildvrt -separate)
- tile:          Generate tiles in separate files from a raster dataset.
                  (enhanced gdal2tiles)
- tpi:           Generate a Topographic Position Index (TPI) map   (gdaldem
TPI)
- tri:           Generate a Terrain Ruggedness Index (TRI) map    (gdaldem
TRI)
- unscale:       Convert scaled values of a raster dataset into unscaled
                 Values.  (gdal_translate -unscale)
- viewshed:      Compute the viewshed of a raster dataset   (gdal_viewshed)
```

`gdal raster tile`

- C++ port of `gdal2tiles.py`
- Enhancements
  - Handle non-Byte data types (UInt16, Float32…)
  - More tiling schemes
    - consecutive overview levels with resolution != 2 (ie NZTM2000)
    - alternative origin
    - different tile size
  - 3x to 6x faster
  - Unified progress report

gdal vector

- clip: Clip a vector dataset *(ogr2ogr -clipsrc / -clipdst)*
- concat: Concatenate vector datasets *(ogr_merge.py)*
- convert: Convert a vector dataset *(ogr2ogr)*
- edit: Edit metadata of a vector dataset *(ogr2ogr -a_srs/-mo)*
- filter: Filter a vector dataset *(ogr2ogr -spat/-where)*
- geom: Geometry operations on a vector dataset *(ogr2ogr …)*
- grid: Create a regular grid from scattered points *(gdal_grid)*
- info: Return information on a vector dataset *(ogrinfo)*
- pipeline: Process a vector dataset
- rasterize: Burns vector geometries into a raster *(gdal_rasterize)*
- reproject: Reproject a vector dataset *(ogr2ogr -t_srs)*
- select: Select a subset of fields *(ogr2ogr -select)*
- sql: Apply SQL statement(s) to a dataset *(ogr2ogr -sql)*

gdal vector geom

```
  - buffer:                Compute a buffer around geometries
  - explode-collections: Explode geometries of type collection
  - make-valid:            Fix validity of geometries of a vector
dataset.
  - segmentize:            Segmentize geometries of a vector dataset.
  - set-type:              Modify the geometry type of a vector dataset.
  - simplify:              Simplify geometries of a vector dataset.
  - swap-xy:               Swap X and Y coordinates of geometries
```

Most above commands correspond to various switches in ogr2ogr.

Note: in GDAL 3.12, the “geom” sub-level will be removed and those commands will be directly available under “gdal vector”

gdal vector grid

Current `gdal_grid` uses a `-a "{alg_name}:key1=val1:key2=val2"` syntax.
Cleaner with sub-commands:

```
  - average:                Create a regular grid from scattered points
                            using moving average interpolation.
  - average-distance:       using the average distance between the grid
                            node (center of the search ellipse) and all
of
                            the data points in the search ellipse.
  - average-distance-points: using the average distance between the
data
                            points in the search ellipse.
  - count:                  using number of points in the search ellipse.
  - invdist:                weighted inverse distance interpolation.
  - invdistnn:              weighted inverse distance interpolation
                            nearest neighbour.
  - linear:                 linear/barycentric interpolation.
  - maximum:                maximum value in the search ellipse.
  - minimum:                minimum value in the search ellipse.
  - nearest:                nearest neighbor interpolation.
  - range:                  difference between the minimum and maximum
                            values in the search ellipse
```

GDALG: generating pipelines

- **Some raster or vector algorithms accept streamed input and generate streamed outputs ⇒ processing pipelines**

```
$ gdal raster pipeline read in.tif ! \
    reproject --dst-crs=EPSG:4326 ! \
    edit --metadata TITLE=my-title ! \
    write out.tif --of COG
```

```
$ gdal vector pipeline ! \
    read in.gpkg ! \
    select fields=name,geometry ! \
    reproject --dst-crs=EPSG:4326 ! \
    geom make-valid ! \
    clip --box=2,49,3,50 ! \
    write out.gpkg --overwrite
```

GDALG: serializing pipelines

- Generalization of GDAL or OGR VRT
- GDALG format just records the command line

```
$ gdal raster pipeline read in.tif ! \
   reproject --dst-crs=EPSG:4326 ! \
   edit --metadata TITLE=my-title ! \
   write out.gdalg.json
```

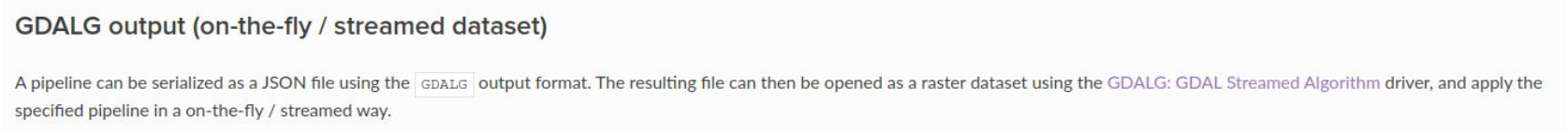
```
$ cat out.gdalg.json
{
  "type":"gdal_streamed_alg",
  "command line":"gdal raster pipeline read --input
in.tif ! reproject --dst-crs EPSG:4326 ! edit
--metadata TITLE=my-title",
  "gdal_version": "3110000"
}
```

GDALG: replaying pipelines

```
$ gdal info out.gdalg.json --of=text
Driver: GDALG/GDAL Streamed Algorithm driver
Files: out.gdalg.json
Size is 16384, 8192
Coordinate System is:
GEOGCRS["WGS 84",
[...]
    ID["EPSG",4326]]
Origin = (-180.000000000000000,90.000000000000000)
Pixel Size = (0.021972656250000,-0.021972656250000)
Metadata:
  TITLE=my-title
Corner Coordinates:
[..]
Band 1 Block=512x128 Type=Byte, ColorInterp=Red
Band 2 Block=512x128 Type=Byte, ColorInterp=Green
Band 3 Block=512x128 Type=Byte, ColorInterp=Blue
```

GDALG: which algorithms?

- **Documentation page of each command indicates if GDALG is supported**
  E.g https://gdal.org/en/latest/programs/gdal_raster_pipeline.html



GDALG output (on-the-fly / streamed dataset)

A pipeline can be serialized as a JSON file using the `GDALG` output format. The resulting file can then be opened as a raster dataset using the GDALG: GDAL Streamed Algorithm driver, and apply the specified pipeline in a on-the-fly / streamed way.

- **Or --help message:**

```
  -f, --of, --format, --output-format <OUTPUT-FORMAT>
Output format ("GDALG" allowed)
```

Bash completion

- **Enabled in Docker images and gdal conda-forge package**
- **Command and subcommand completion**

```
$ gdal <TAB><TAB>
convert   driver  info   mdim   pipeline   raster  vector
    vsi
```

- **Option name completion**

```
$ gdal raster convert --<TAB><TAB>
--append              --creation-option  --if              --input-format
--oo                  --output           --overwrite       --co
--format              --input            --of              --open-option
--output-format       --progress
```

Bash completion

- **Option value completion**

```
$ gdal raster convert --of=<TAB><TAB>
AAIGrid         DDS             FITS            GTiff
AVIF            DTED            GeoRaster       GTX
BAG             ECW             GIF             HDF4Image
BASISU          EHdr            GPKG            HEIF
[...]
```

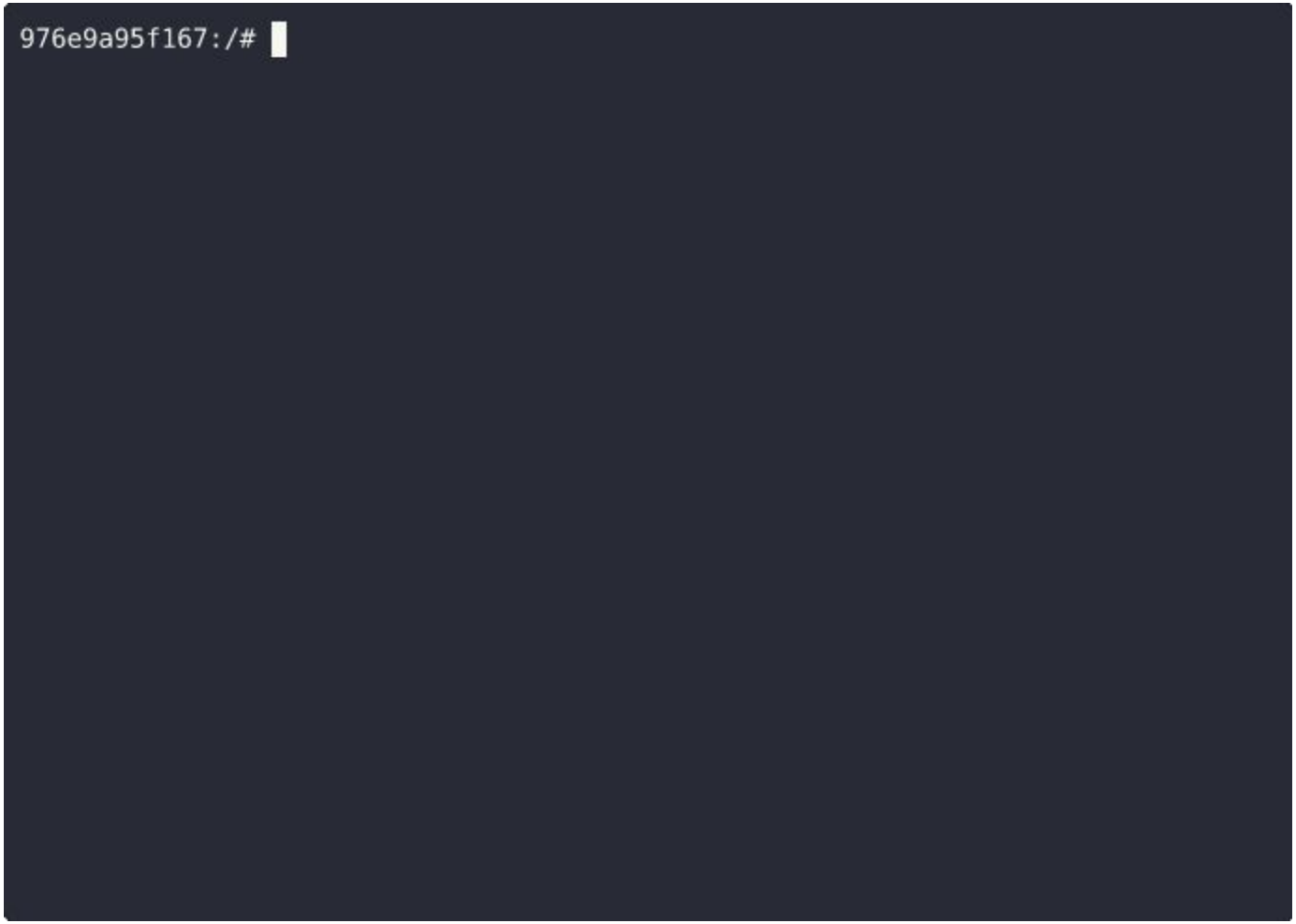
- **Driver-dependent creation-option name**

```
$ gdal raster convert --of=GTIFF --co=<TAB><TAB>
ALPHA=                          ENDIANNESS=                     JXL EFFORT=
BIGTIFF=                        GEOTIFF KEYS FLAVOR=            JXL LOSSLESS=
BLOCKXSIZE=                     GEOTIFF_VERSION=                LZMA_PRESET=
[...]
```

- **creation-option value suggestion: enumeration**

```
$ gdal raster convert --of=GTIFF --co COMPRESS=
CCITTFAX3 CCITTFAX4     CCITTRLE   DEFLATE   JPEG    JXL
LERC      LERC DEFLATE  LERC_ZSTD    LZMA      LZW     NONE
PACKBITS  WEBP          ZSTD
```

Bash completion



Bash completion

- **creation-option value suggestion: min/max**

```
$ gdal raster convert --of=GTIFF --co JPEG QUALITY=<TAB><TAB>
##                        validity range: [1,100]
```

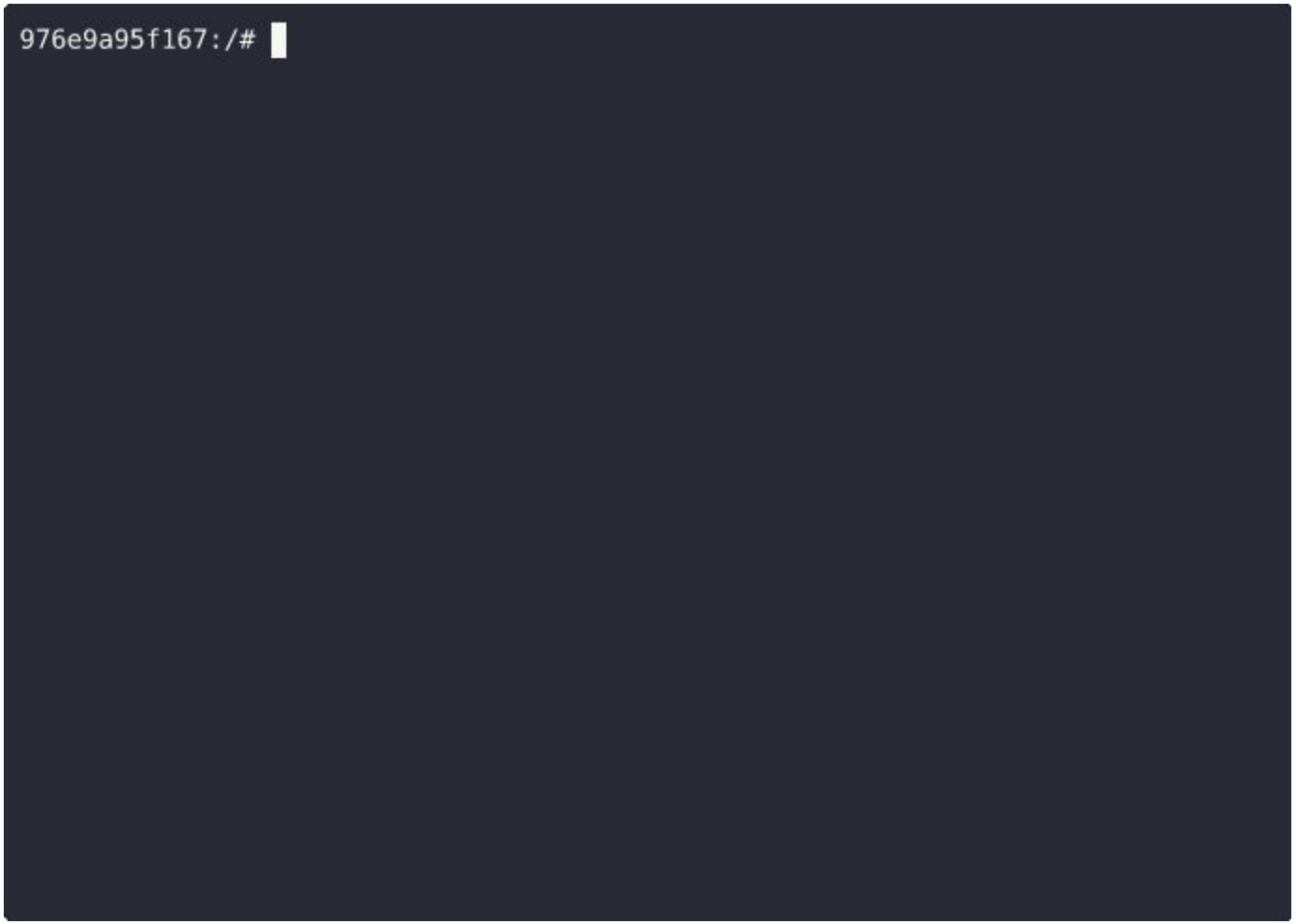
- **CRS code completion**

```
$ gdal raster reproject --dst-crs=EPSG:326<TAB><TAB>
32600 -- WGS 84 / UTM grid system (northern hemisphere)
32601 -- WGS 84 / UTM zone 1N
32602 -- WGS 84 / UTM zone 2N
[...]
```

- **VSI filenames completion !**

```
$ gdal vsi list -lR --of=text /vsis3/my_bucket/World<TAB><TAB>
⇒
$ gdal vsi list -lR --of=text /vsis3/my_bucket/WorldCRS84Quad
```

Bash completion (/vsi)



Python usage (1 / 2)

- **Getting argument names:**

```
>>> gdal.Algorithm("raster", "convert").GetArgNames()
['output-format', 'open-option', 'input-format', 'input',
'output', 'creation-option', 'overwrite', 'append']
```

- **Executing an algorithm:**

```
>>> gdal.Run("raster", "convert", input="in.tif",
              output="out.tif", output_format="COG",
              overwrite=True)
```

or

```
>>> alg = gdal.Algorithm("raster”", "convert")
>>> alg["input"] = "in.tif"
>>> alg["output"] = "output.tif"
>>> alg["output-format"] = "COG"
>>> alg["overwrite"] = True
>>> alg.Run()
>>> alg.Finalize() # fully close opened datasets
```

Python usage (2 / 2)

- **Getting output information as JSON**

```
>>> alg = gdal.Run("raster", "info", input="byte.tif")
>>> info_as_dict = alg.Output()
```

- **Getting output information as a GDAL dataset object**

```
>>> with gdal.Run("raster reproject",
              input=src_ds, output_format="MEM",
              dst_crs="EPSG:4326"}) as alg:
>>>    values = alg.Output().ReadAsArray()
```

⇒ **See https://gdal.org/programs/gdal_cli_from_python.html**

--json-usage

ex: gdal raster convert


```
{
  "name":"convert",
  "full_path":[ "gdal", "raster","convert" ],
  "description":"Convert a raster dataset.",
  "url":"https://gdal.org/programs/gdal_raster_convert.html",
  "input_arguments":[
   {
      "name":"output-format",
      "type":"string",
      "description":"Output format",
      "required":false,
      "metadata":{
            "required_capabilities":[
              "DCAP_RASTER", "DCAP_CREATECOPY" ]
      }
   },
   { "name":"open-option", [...] },
   { "name":"input-format", [...] },
   {
      "name":"input",
      "type":"dataset",
      "description":"Input raster dataset",
      "required":true,
      "Dataset_type":[ "raster" ],
      "input_flags":[ "name", "dataset" ],
      ]
```

```
   {
     "name":"creation-option",
     "type":"string_list",
     "description":"Creation option",
     "metavar":"<KEY>=<VALUE>",
     [...]
     "max_count":2147483647,
   },
   {
    "name":"overwrite",
     [...]
     "mutual_exclusion_group":"overwrite-append"
   },
   {
     "name":"append",
     [...]
     "mutual_exclusion_group":"overwrite-append"
    }
  ],
  "output_arguments":[],
  "input_output_arguments":[
  {
      "name":"output",
      "type":"dataset",
      "description":"Output raster dataset",
      "required":true,
      "dataset_type":[ "raster" ],
      "input_flags":[ "name" ],
      "output_flags":[ "dataset" ]
     }
  ]
}
```


GDAL

gdal vsi (1/2)

- `gdal vsi list -lR --of=text /vsis3/bucket`

```
---------- 1 unknown unknown         2740 2024-03-25 23:43 9563.tif
---------- 1 unknown unknown        18164 2017-09-22 11:37 LICENSE.TXT
---------- 1 unknown unknown       530674 2018-11-08 15:34 NEWS
---------- 1 unknown unknown       269811 2019-08-07 20:47 REANALYSIS_1999217.hdf
d--------- 1 unknown unknown            0 2020-11-27 12:35 WorldCRS84Quad
d--------- 1 unknown unknown            0 2020-11-27 12:35 WorldCRS84Quad/0
d--------- 1 unknown unknown            0 2020-11-27 12:35 WorldCRS84Quad/0/0
---------- 1 unknown unknown         1077 2020-11-27 12:35 WorldCRS84Quad/0/0/0.tif
d--------- 1 unknown unknown            0 2020-11-27 12:35 WorldCRS84Quad/1
d--------- 1 unknown unknown            0 2020-11-27 12:35 WorldCRS84Quad/1/0
---------- 1 unknown unknown         2220 2020-11-27 12:35 WorldCRS84Quad/1/0/0.tif
d--------- 1 unknown unknown            0 2020-11-27 12:35 WorldCRS84Quad/2
d--------- 1 unknown unknown            0 2020-11-27 12:35 WorldCRS84Quad/2/0
---------- 1 unknown unknown         3706 2020-11-27 12:35 WorldCRS84Quad/2/0/1.tif
---------- 1 unknown unknown      5135852 2020-09-07 13:23 bmng_out.tif
---------- 1 unknown unknown          372 2021-04-27 14:17 byte.png
---------- 1 unknown unknown         1797 2025-05-02 13:23 byte.tif
---------- 1 unknown unknown          351 2025-01-27 19:24 byte.tif.aux.xml
```

- `--of=json` output also available

gdal vsi (2/2)

- `gdal vsi` **`copy`** `-r /vsis3/my_bucket/my_dir /vsigs/dest_bucket`
  ~ analog to UNIX `cp`
- `gdal vsi` **`remove`** `-r /vsis3/my_bucket/my_dir`
  ~ analog to UNIX `rm`
- `gdal vsi` **`move`** `/vsigs/bucket/foo /vsigs/bucket/bar`
  ~ analog to UNIX `mv` to move or rename (including between different object storages, through local computer!)
- `gdal vsi` **`sync`** `-r local_dir/ /vsis3/bucket/tgtdir`
  - ~ analog to UNIX `rsync`
  - Several synchronization strategies: timestamp (default), ETag/md5sum, or always overwrite

gdal driver

- `$ gdal driver --help`

```
Usage: gdal driver <SUBCOMMAND> [OPTIONS]
where <SUBCOMMAND> is one of:
  - gpkg:         Command for GPKG driver specific operations.
  - gti:          Command for GTI driver specific operations.
  - openfilegdb: Command for OpenFileGDB driver specific operations.
  - pdf:          Command for PDF driver specific operations.
```

- `gdal driver gti create`: **Create an index of raster datasets compatible with the GDAL Tile Index (GTI) driver (extension of gdal raster index with GTI-specific part of gdaltindex)**
- `gdal driver gpkg repack`: **Repack/vacuum in-place a GeoPackage dataset**
- `gdal driver openfilegdb repack`: **Repack a FileGeoDatabase dataset**

Extensible framework

- Plug your own subcommands into `gdal`
  - Register a GDAL plugin shared object
- Out-of-tree drivers can register `gdal driver` algorithms

Future of `gdal_translate`, `ogr2ogr` ?

- 25+ years of history
- No breaking changes to main CLIs
- New capabilities -> `gdal` commands
- Migration and decommission schedule of `gdal`-ported Python-based CLIs

- Community feedback needed!

`gdal` Maturation Schedule

- First release of `gdal` – 3.11 (May 13, 2025)
- Please share feedback!

- No promise of subcommands, options, or GDALG compatibility until ~3.16.0
- Some utilities not ported yet (~90%)
- Some esoteric options not yet exposed
  - Make ticket traffic to inform us of demand

(Main) contributors (code, review, design, doc, testing)

Even Rouault

Dan Baston

Alessandro Pasotti

Jukka Rahkonen

Howard Butler

Sean Gillies

- Gold level:

planet.

esri

Microsoft

- Silver level:

Google

Safe Software

- Bronze level:

- Supporter level:

Dynamic Graphics, Inc. Umbra Vortex f.d.c. Route4Me, Inc. PIX4D Satelligence Space Intelligence Myles Sutherland T-Kartor Kaplan Open Source Consulting

Regrid Phoenix LiDAR Systems, LLC

GDAL

Thank you sponsors!

Sponsorship Program Outlook

- 💰 outlook is poor! 😰
  - Business cycle downturn
  - Sustainment is difficult
  - People move on
- Maintenance program outlook is ✅
  - We haven't overspent
  - Conservative initiatives
- Capacity to fund more maintenance 📈
  - Bootstrapping contributors
  - Supporting dependencies
    - Sphinx / Breathe / RTD
    - PROJ
    - GEOS

- Gold level:

planet.

esri

Microsoft

- Silver level:

Google

Safe Software

- Bronze level:

- Supporter level:

Dynamic Graphics, Inc. Umbra Vortex f.d.c. Route4Me, Inc. PIX4D Satelligence Space Intelligence Myles Sutherland T-Kartor Kaplan Open Source Consulting

Regrid Phoenix LiDAR Systems, LLC

GDAL

Thank you sponsors!


GDAL

Q&A session (1/10)

- **Q: Are there be plans to add DuckDB as a supported SQL dialect for `gdal vector sql` in the future?**
  A: Not directly, but you can already use the OGR ADBC driver with libduckdb. See https://gdal.org/stable/drivers/vector/adbc.html
  Here's an example:

```
ogrinfo -ro ADBC: -oo ADBC_DRIVER=libduckdb -oo SQL="select
* from
read_parquet(\"s3://overturemaps-us-west-2/release/2024-12-1
8.0/theme=places/type=place/*\", filename=true,
hive_partitioning=1)  where
st_dwithin_spheroid(geometry,ST_POINT( -72.1440, 43.6406 ),
500)=true and bbox.xmin BETWEEN -73 AND -72 AND bbox.ymin
BETWEEN 43 AND 44 limit 1" -al
-oo PRELUDE_STATEMENTS="LOAD SPATIAL"
-oo PRELUDE_STATEMENTS="LOAD PARQUET"
-oo PRELUDE_STATEMENTS="load httpfs"
```

Q&A session (2/10)

- **Q: Is there `gdal vector tiles` ?**
  A: Not currently. But you may use the existing MBTiles, PMTiles or PBF vector drivers to generate ones.

- **Q: Will the pipeline functionality be able to take advantage of multi-cpus in case of transformations on many files?**
  A: If one of the processing stage supports multi threading (e.g. raster reproject), then yes. Pipeline execution itself is sequential.
  Another option is to do this via xargs/parallel/dask/power shell if it's across multiple files

Q&A session (3/10)

- **Q: Can you have two `read`s in `gdal vector pipeline` and/or does it make joins easier?**
  A: No, “read” reads a single file. You can also use the “concat” stage as the initial one to “merge” together multiple files (although not joins)

- **Q: Why not GDAL 4.x with this new CLI?**
  A: There will be lots of changes over the subsequent releases. We haven’t decided when to change to 4.0. Ticket https://github.com/OSGeo/gdal/issues/8440 contains potential ideas for an hypothetical 4.0 release.

Q&A session (4/10)

- **Q: Will there be refactoring for the Python API with the same design patterns? And if so, how can we contribute there?**
  A: To be debated. But you can access the commands/algorithms of the new "gdal" CLI in a nice way from Python as shown in slides 34/35

- **Q: If the command does not support the new pipeline what is the error msg?**
- **A: Example:**

```
$ gdal raster pipeline read byte.tif ! calc
```

```
ERROR 1: pipeline: unknown step name: calc
```

Q&A session (5/10)

- **Q: Did I understand well, that is `gdalg.json` file could be used to replace VRT files?**
  A: Not all VRT files, but VRTs generated by `gdal_translate` or `gdalwarp` can be replaced by `.gdalg.json` files

- **Q: Why is there a spatial `mdim` category for commands? Aren't those vector or raster commands?**

  A: The multidimensional API is the 3rd (and less known) data model of GDAL besides raster and vector. See https://gdal.org/user/multidim_raster_data_model.html for more details

Q&A session (6/10)

- **Q: Where can we find which commands support pipelines?**
  A: https://gdal.org/programs/gdal_vector_pipeline.html
  & https://gdal.org/programs/gdal_raster_pipeline.html

- **Q: Am I out-of-the-loop, or is there a GDAL development roadmap to know a bit about what further is coming (and how we might contribute)?**
  A: These are the RFCs
  https://gdal.org/en/latest/development/rfc/index.html

Q&A session (7/10)

- **Q: Is it possible to convert the NoData value to NaN in the new version?**
  A: Yes, you can do that with `gdal raster reclassify` or `gdal raster reproject`

- **Q: Managing the credentials for GDAL vsis3 is still done the same ways when using `gdal` ?**
  A: Yes

- **Q: How many hours did go into this and how much of that was funded?**
  A: Probably around 600 hours, nearly all of them funded through the GDAL Sponsorship Program and a NASA grant

Q&A session (8/10)

- **Q: Is the internal functionality exactly the same or have any changes?**
  A: It depends. Some of the Python programs have been rewritten in C++. There have been improvements all over.

- **Q: It was mentioned the GDAL CLI could be extended, is there already some documentation on this?**
  A: We probably lack a tutorial. The main entry point would be the GDALGlobalAlgorithmRegistry::DeclareAlgorithm() method:
  https://github.com/OSGeo/gdal/blob/85818e4c2b57677a179e5642ae605ca9e6ea5b66/gcore/gdalalgorithm.h#L3101

Q&A session (9/10)

- **Q: Is there a way to donate at the individual level?**
  A: Yes, see https://numfocus.org/donate-to-gdal. But making the case for sponsoring within your organization would be even more efficient way to resource the project.

- **Q: What's your take on using Rust for future GDAL-related development—any clear benefits or interest in the community?**
  A: Tricky topic. Given what GDAL does (reading arbitrary files), using a memory safe language would make a lot of sense. But currently the C++ / Rust interoperability is still lacking mature and efficient tooling. That said, you can use GDAL from Rust today.

Q&A session (10/10)

- **Q: Would love to see more examples and tutorials grow, are there any overhauls coming to showcase more work done there? Can people outside of the sponsors contribute to document examples/tutorials and where to do so?**
  A: Yes, we welcome them. Just open a ticket!

Backup Slides

Cheat sheet: raster commands (1 / 2)

- https://gdal.org/en/latest/programs/migration_guide_to_gdal_cli.html
- **Format conversion**

```
gdal_translate in.tif out.tif -of COG -co COMPRESS=ZSTD
```

⇒ `gdal raster convert in.tif out.tif --of=COG --co=COMPRESS=ZSTD`

- **Resizing**

```
gdal_translate in.tif out.tif -outsize 50% 50%
```

⇒ `gdal raster resize in.tif out.tif --size=50%,50%`

- **Clipping**

```
gdal_translate in.tif out.tif -projwin 2 50 3 49
```

⇒ `gdal raster clip in.tif out.tif --bbox=2,49,3,50`

- **Reprojecting**:

```
gdalwarp in.tif out.tif -t_srs EPSG:4326
```

⇒ `gdal raster reproject in.tif out.tif --dst-crs=EPSG:4326`

Cheat sheet: raster commands (2 / 2)

- **Line contouring**

```
gdal_contour -i 100 in.tif out.gpkg
```

⇒ `gdal raster contour --interval=30 in.tif out.gpkg`

- **Hillshade creation**

```
gdaldem hillshade -z 30 -s 111120 n043w100.dt1 out.tif
```

⇒ `gdal raster hillshade -z 30 n043w100.dt1 out.tif`

- **Index creation**

```
gdaltindex out.shp *.tif
```

⇒ `gdal raster index *.tif out.shp`

- **Tile creation (in WebMercator)**

```
gdal2tiles --num-processes=12 -r lanczos in.tif out_dir --xyz
```

⇒ `gdal raster tile -j 12 -r lanczos in.tif out.tif`

Cheat sheet: vector commands (1 / 2)

- **Format conversion**
  `ogr2ogr out.gpkg in.shp`

⇒ `gdal vector convert in.shp out.gpkg`

- **Clipping**
  `ogr2ogr out.gpkg in.shp -clipsrc 2 49 3 50`

⇒ `gdal vector clip in.shp out.gpkg --bbox=2,49,3,50`

- **Reprojecting**
  `ogr2ogr out.gpkg in.shp -t_srs EPSG:4326`

⇒ `gdal vector reproject in.shp out.gpkg --dst-crs=EPSG:4326`

- **Selecting subset of fields**
  `ogr2ogr out.gpkg in.shp -select name`

⇒ `gdal vector select in.shp out.gpkg --fields=name,_ogr_geometry_`

Cheat sheet: vector commands (2 / 2)

- **Extracting subset of data through SQL**

```
ogr2ogr out.gpkg my.shp -sql "SELECT * FROM my WHERE pop > 5e6"
```

⇒

```
gdal vector sql in.shp out.gpkg --sql "SELECT * FROM my WHERE pop > 5e6"
```

- **Converting polygons to multipolygons**

```
ogr2ogr out.gpkg my.shp -nlt MULTIPOLYGON
```

⇒

```
gdal vector geom set-type in.shp out.gpkg --geometry-type MULTIPOLYGON
```

- **Stacking all input shapefiles into separate layers of a GeoPackage**

```
ogrmerge -f GPKG -o merged.gpkg *.shp
```

⇒

```
gdal vector concat --mode=stack *.shp merged.gpkg
```

gdal raster calc

- C++ analog to `gdal_calc.py` Python script
- Evaluates expressions using muparser library
- Lazy evaluation using VRT/GDALG

```
$ SRC="/vsis3/sentinel-cogs/sentinel-s2-l2a-cogs/26/S/PG/2025/
       4/S2C_26SPG_20250424_0_L2A"

$ gdal raster calc -i "NIR=${SRC}/B08.tif" \
                   -i "R=${SRC}/B04.tif" \
                   -o NDVI.vrt \
                   --calc "(NIR-R)/(NIR+R)"

$ echo 640000 4187500 | gdal raster pixel-info --position-crs dataset
NDVI.vrt | jq .features[].properties.bands[].raw_value
0.8737033666969722
```

gdal raster reclassify

- **Reclassify a raster using input ranges**

```
$ gdal raster reclassify \
    -i gradient.tif \
    -o gradient_cat.vrt \
    --mapping "[1,3]= 101; [4, 5)= 102; 7=102; NO_DATA=103;
DEFAULT=NO_DATA"
    --output-data-type Byte
```