



niva

Readable geoprocessing for everyone

Spatial analysis a non-programmer can write, reproduce, and share.

Easy wins every time

Decision-maker & analyst briefing

Spatial questions are everywhere

The people who need the answers — analysts, scientists, planners — **mostly aren't programmers.**

Yet to **automate** the work — to make it repeatable and shareable — today's tools demand that they write code. So the work waits, or doesn't happen.

niva · Easy wins every time

THE PROBLEM

Automating QGIS means writing PyQGIS

To script even simple geoprocessing today, you must:

- **Initialize a `QgsApplication`** — boilerplate before any work begins
- **Memorize algorithm IDs** — `native:buffer`, `native:dissolve`, `native:clip`
- **Build ALL_CAPS parameter dicts** — and juggle `TEMPORARY_OUTPUT`
- **Thread each output into the next** — by hand, step after step

That's a programming task — out of reach for the GUI-first analysts who need it most, and tedious even for those who can.

Automation shouldn't require a programmer

Without niva — write PyQGIS

A programming task: init, algorithm IDs, ALL_CAPS dicts, temp outputs, threading

```
from qgis.core import QgsApplication
import processing
qgs = QgsApplication([], False)
qgs.initQgis()
processing.run("native:buffer", {
    "INPUT": "roads.gpkg",
    "DISTANCE": 100,
    "DISSOLVE": True,
    "OUTPUT": "TEMPORARY_OUTPUT",
})["OUTPUT"]
# ...now thread that into clip...
# ...then into save... repeat per step.
# memorize IDs · params · plumbing
→ out of reach for GUI-first analysts
```



With niva — one readable line

```
load roads.gpkg
| buffer 100 dissolve
| clip city.gpkg
| save roads_local.gpkg
```

- Readable — review it like a sentence
- Reproducible — same flow, same result
- Shareable — one line, not a script
- Full QGIS power underneath

Easy wins every time.

THE PRODUCT

Meet niva

A concise, readable **text-pipeline grammar** for QGIS geoprocessing — for people who don't want to write PyQGIS.

```
load roads.gpkg | buffer 100 dissolve | clip city.gpkg | save roads_local.gpkg
```

A whole pipeline on one line — that a non-programmer can **write and read** — running on QGIS's own Processing algorithms underneath.

Easy wins every time.

niva · Easy wins every time

AUDIENCE

Who niva is for

Decision-makers

Faster answers without hiring scarce GIS programmers. Work that's auditable, reproducible, and shareable across the org.

Data scientists (non-programmers)

Already use Marimo for repeatable, shareable analysis. Now add geoprocessing — in the same readable, reactive workflow.

Downstream consumers

Explore the results in QGIS, in Marimo, or as static files — whichever fits, no GIS expertise required.

niva · Easy wins every time

It fits the workflow you already have

Marimo is how your team makes analysis repeatable and shareable. niva runs right inside it:

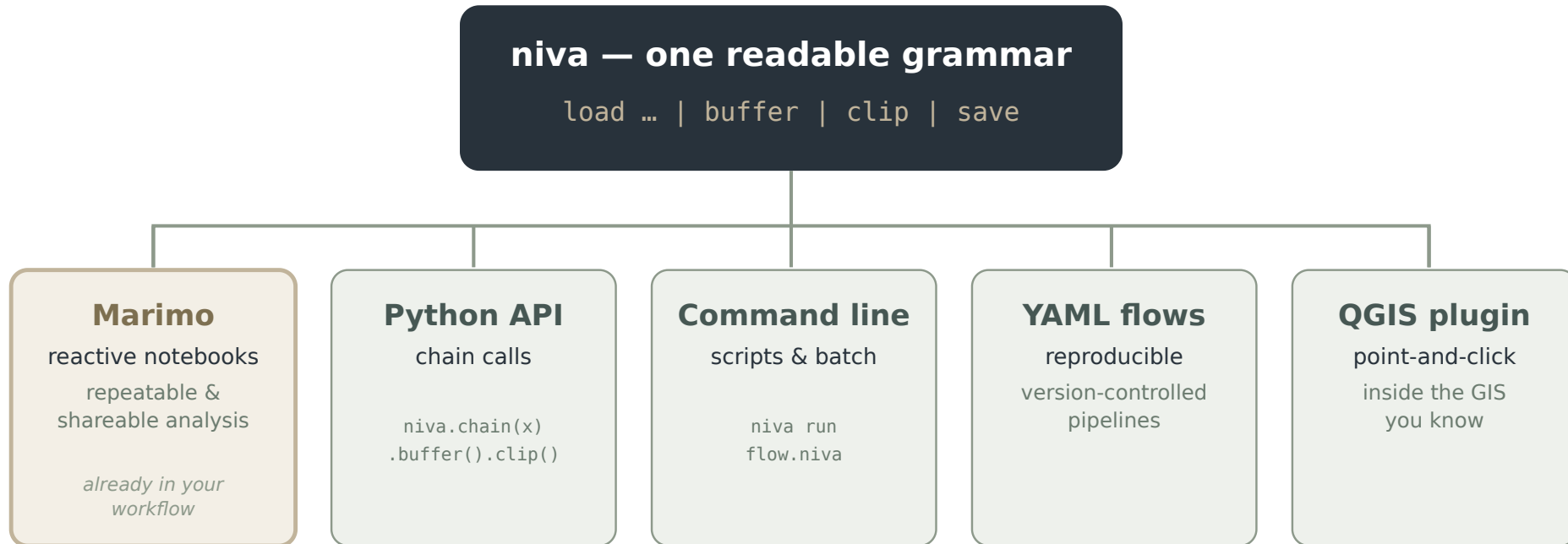
- **Write niva in a Marimo cell** — beside your Python and charts
- **Reactive** — change an input, the map and tables update live
- **Repeatable** — the notebook is the analysis; re-run any time, same result
- **Shareable** — hand someone the notebook, or export it; no setup ritual

No new tool to adopt — niva meets your people where they already work.

niva · Easy wins every time

FLEXIBLE BY DESIGN

One grammar, many surfaces



Meet people where they are — **the same grammar everywhere.**

...and it all runs on QGIS's own Processing algorithms underneath.

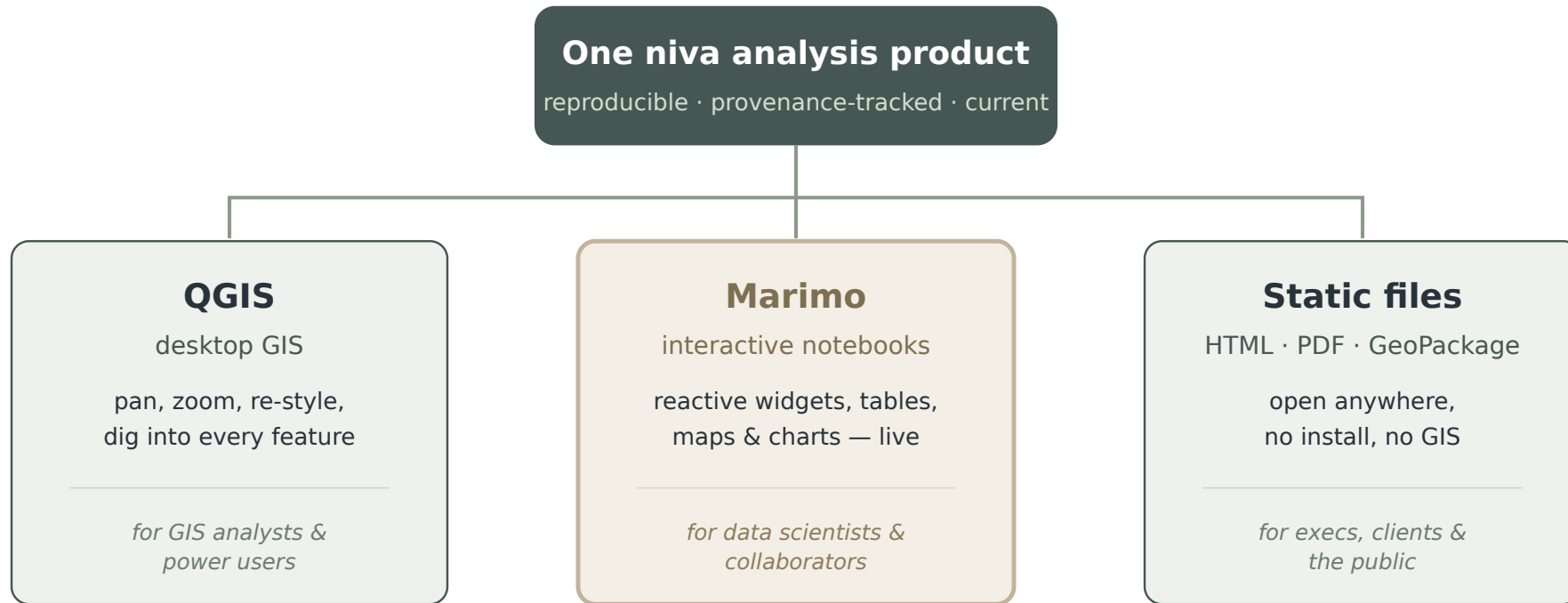
Reproducible & shareable — by design

The value decision-makers care about most: results you can re-run, audit, and defend.

- **Pipelines are the record** — the one-line flow (or YAML) *is* the documentation
- **Provenance as a byproduct** — every step auto-recorded as lineage metadata
- **Assess your inputs** — profile incoming data for quality before you trust it
- **Version-controlled flows** — diff and review analysis like any other text

FOR EVERY STAKEHOLDER

Produce once — explore it anywhere

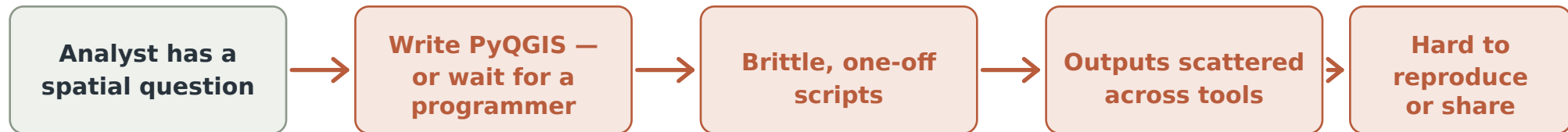


Produce once — **everyone explores it the way that suits them.**

The workflow today

Today: automation means programming

The analyst who needs the answer can't get it without code — or someone who writes it.



The cost

- Automation is gated on scarce programming skill — everyday work waits in a queue.
- Results are siloed and stale; stakeholders get one-off exports they can't explore or trust.

The workflow with niva

With niva: the analyst is self-sufficient

Write a readable pipeline in the tools you already use — reproducible and shareable by design.



The payoff

- No programming bottleneck — the person with the question does the work, same day.
- One reproducible, audited flow; everyone works from the same current analysis, in their tool of choice.

THIN, NOT LIMITING

All of QGIS — in plain language

878

Processing algorithms

406

expression functions

SQL

SpatiaLite & PostGIS

niva is a **thin wrapper** — friendly verbs mapped onto QGIS's own algorithms, not a re-implementation of GIS. The full power of the world's leading open-source GIS, in one readable line.

niva · Easy wins every time

Every data type QGIS reads

Point clouds

- ✓ each / show / catalog see LAS · LAZ · COPC · E57 · ...
- ✓ Friendly dtm · dsm · hag — raw LiDAR → raster

Raster · mesh · vector

- ✓ Discovery spans **all** GDAL formats + MDAL mesh
- ✓ One inventory: catalog data/ to=report.md

A folder of LiDAR tiles is just another pipeline input:

```
each "tiles/*.las" | dtm resolution=1 | save "dtm/{name}.tif"
```

From a point cloud to a watershed

One bare-earth DTM → a full hydrologic stack, in plain lines:

```
each "*.las" | dtm resolution=1 | save "dtm/{name}.tif"  
run gdal:merge INPUT="dtm/*.tif" OUTPUT=dtm.tif  
run grass:r.watershed elevation=dtm.tif accumulation=accum.tif drainage=dir.tif basin=basin.tif  
run grass:r.to.vect input=basin.tif type=2 output=catchments.gpkg  
run saga:ta_channels:5 DEM=dtm.tif ORDER=strahler.tif SEGMENTS=streams.gpkg BASINS=basins.gpkg
```

Flow direction · flow accumulation · **vector catchments** · **Strahler-ordered streams** — GRASS & SAGA, one grammar.

niva · Easy wins every time

FUSE ANYTHING

Terrain, imagery & features — together

Multispectral

- ✓ Landsat · Sentinel-2 · NAIP → NDVI
- ✓ Zonal stats onto parcels — greenness per lot

LiDAR → vector

- ✓ Contours · slope zones from the DTM
- ✓ Elevation & canopy per building footprint

Rasters become lines and polygons; every vector product lands in one GeoPackage.

niva · Easy wins every time

Why niva, why now

- **QGIS is the de-facto open GIS** — ubiquitous, trusted, free; the foundation we build on
- **Reproducible analysis is the norm** — Marimo and notebooks are how modern teams work
- **The automation gap is real & unserved** — GUI analysts still can't automate without code
- **Open and clean-room** — GPLv3, built on PyQGIS; no proprietary lock-in to fear

niva · Easy wins every time

What niva delivers

Speed

Answers the same day — no queue behind a programmer.

Capacity

Existing analysts automate their own work.

Trust

Reproducible, provenance-tracked, auditable results.

Reach

Share to QGIS, Marimo, or static files — for anyone.

Lower cost

Less custom scripting, less specialized hiring.

No lock-in

Open source, on open foundations.

niva · Easy wins every time

What's working today

niva ships as a **QGIS plugin** (v0.62) and on **PyPI**, and runs real analysis end-to-end on QGIS's own algorithms:

Grammar & engine

- ✓ Readable lexer + parser → pipeline stages
- ✓ Pipe-chaining engine: layer handles, lineage
- ✓ PyQGIS backend — runs real geoprocessing

Verbs & algorithms

- ✓ 48 alias verbs + built-ins (vector · raster · point cloud)
- ✓ `sql @conn` — SELECT → layer, and server-side writes
- ✓ `run` — reach ANY of QGIS's 878 algorithms

Every data type

- ✓ Vector · raster · **point cloud** · **mesh** discovery
- ✓ LiDAR: `dtm / dsm / hag` · GRASS + SAGA hydrology
- ✓ Landsat / Sentinel-2 / NAIP fusion → NDVI

Data, cartography & delivery

- ✓ PostGIS & SpatiaLite — read · write · analyse
- ✓ `project / style · figure / map · show / catalog`
- ✓ Plugin · CLI · Python · LSP · 660+ tests · full docs

niva · Easy wins every time

The road ahead

- **v0.1 – 0.62 — Shipped** · available now, on PyPI

Grammar · 48 verbs + raster · `run` → 878 algorithms · point clouds + mesh + all-types discovery · GRASS/SAGA hydrology · multispectral fusion · PostGIS read·write·analyse · project / style · `figure` / `map` · LSP · QGIS plugin + CLI + Python · provenance · full docs

- **v1.0 — Stable release**

Grammar freeze (SemVer) · worked Marimo–QGIS integration

- **v2.0 — Power features**

Named intermediates & variables · SQL-driven quality rules & constraints

- **v2.x — Service mode**

Service / daemon mode · richer layout & symbology export



Let your analysts do the analysis.

Bring readable, reproducible geoprocessing to the people who need it.

Easy wins every time.

github.com/johnzastrow/niva · Let's run a pilot.