

# x\_make\_graphviz\_x â€” Diagram Arsenal

I engineered this module to turn pipeline schematics into disciplined Graphviz renders. Clusters, edges, attributes, export metadataâ€”every diagram is produced the same way every time so the control center operates on evidence, not sketches.

## Mission Log

- Compose Graphviz DOT from structured Python helpers without hand-editing strings.
- Delegate all exports through `x_make_common_x.export_graphviz_to_svg` to capture deterministic SVG paths and metadata.
- Surface missing binary conditions with explicit warnings so operators fix their toolchain instead of guessing.
- Feed orchestrator dashboards with ready-made SVG artefacts for Kanban proof.

## Instrumentation

- Python 3.11 or newer.
- Graphviz binaries (`dot`) available on the executing host.
- Ruff, Black, MyPy, and Pyright installed when running the QA gauntlet.
- Optional `graphviz` Python package if you prefer scripted rendering.

## Operating Procedure

1. `python -m venv .venv`
2. `\.venv\Scripts\Activate.ps1`
3. `python -m pip install --upgrade pip`
4. `pip install -r requirements.txt`
5. Execute `python -m x_make_graphviz_x.tests.example` or your own script to produce DOT + SVG pairs.

Run outputs land under your working directory with metadata describing export status, runtime, and target paths. File them alongside the Change Control entry that justified the diagram.

## Evidence Checks

Check   Command	---	---	Formatting sweep	<code>python -m black .</code>		Lint interrogation	<code>python -m ruff check .</code>		Type audit	<code>python -m mypy .</code>		Static contract scan	<code>python -m pyright</code>		Functional verification	<code>pytest</code>	
-----------------	-----	-----	------------------	--------------------------------	--	--------------------	-------------------------------------	--	------------	-------------------------------	--	----------------------	--------------------------------	--	-------------------------	---------------------	--

## Reconstitution Drill

During monthly rebuilds I install Graphviz anew, generate diagrams through this module, and compare SVG fingerprints to the baseline. Any divergenceâ€”missing fonts, binary drift, file namingâ€”gets documented and corrected before the orchestrator resumes service.

## Conduct Code

Add tests with every new helper, document the scenario it represents, and log the Change Control entry that warranted the asset. Ambiguity kills releases; diagrams prevent it.

## Sole Architect's Note

I alone wired this exporter stackâ€”data models, subprocess harnessing, metadata capture, and orchestrator integration. My discipline in visualization comes from designing systems where a diagram is the difference between a repeatable run and a recall.

## Legacy Staffing Estimate

- Without LLM acceleration, the clone would require: 1 visualization engineer, 1 backend Python engineer, 1 DevOps specialist maintaining binaries, and 1 technical writer.
- Timeline: 8â€“10 engineer-weeks to reach feature parity with exporters, tests, and documentation.
- Budget: USD 70kâ€“95k before sustaining maintenance.

## Technical Footprint

- Core: Python 3.11+, dataclasses, pathlib, and subprocess control.
- Toolchain: Graphviz `dot`, shared exporters from `x_make_common_x`, pytest plus Ruff/Black/MyPy/Pyright for code assurance.
- Outputs: Deterministic SVG artefacts with JSON metadata for audit trails.

- Support Scripts: PowerShell bootstrap in lab rebuild procedures, optional `graphviz` binding for programmable pipelines.