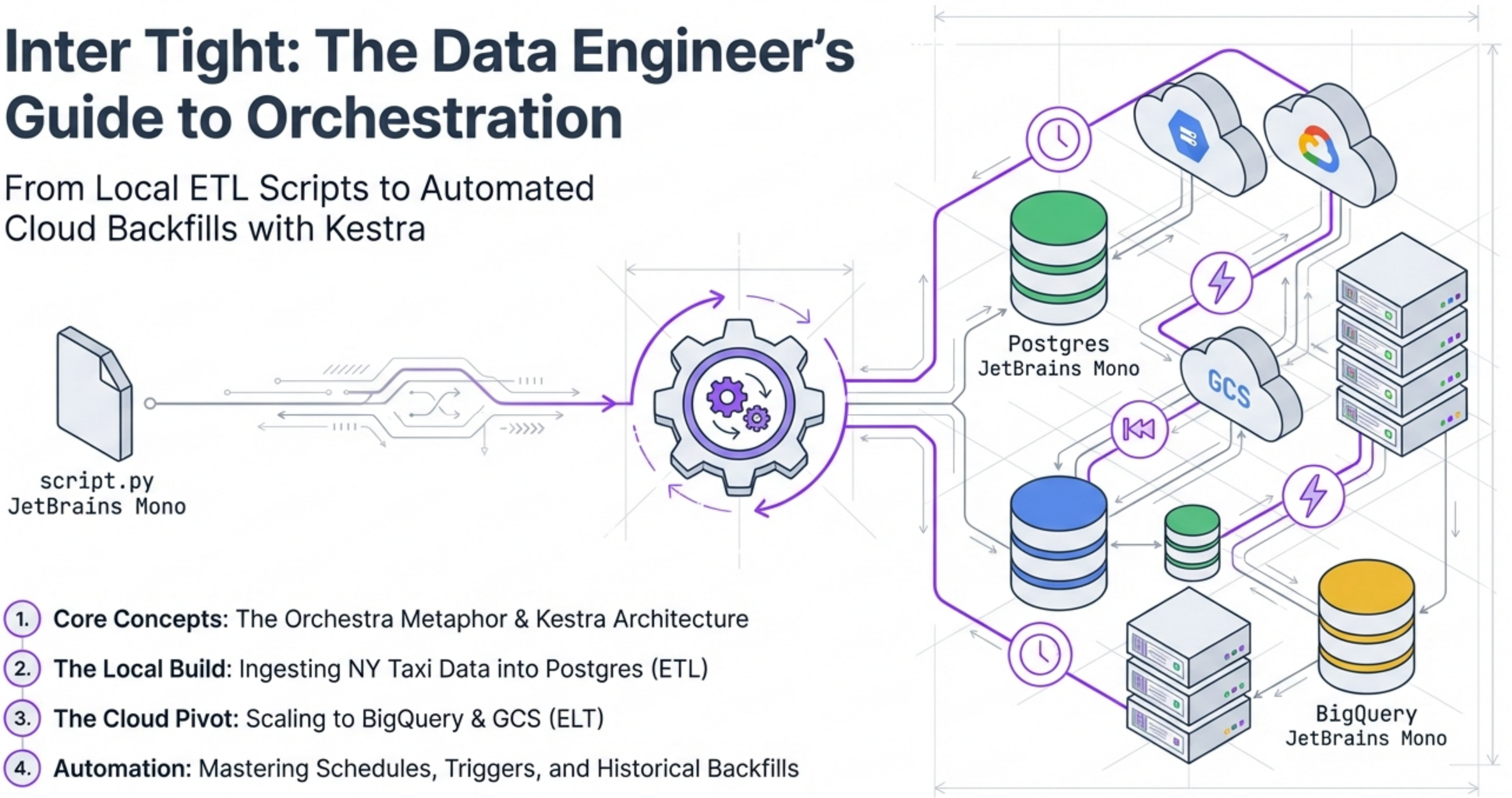
Inter Tight: The Data Engineer's Guide to Orchestration

From Local ETL Scripts to Automated Cloud Backfills with Kestra



1. **Core Concepts:** The Orchestra Metaphor & Kestra Architecture
2. **The Local Build:** Ingesting NY Taxi Data into Postgres (ETL)
3. **The Cloud Pivot:** Scaling to BigQuery & GCS (ELT)
4. **Automation:** Mastering Schedules, Triggers, and Historical Backfills

NotebookLM

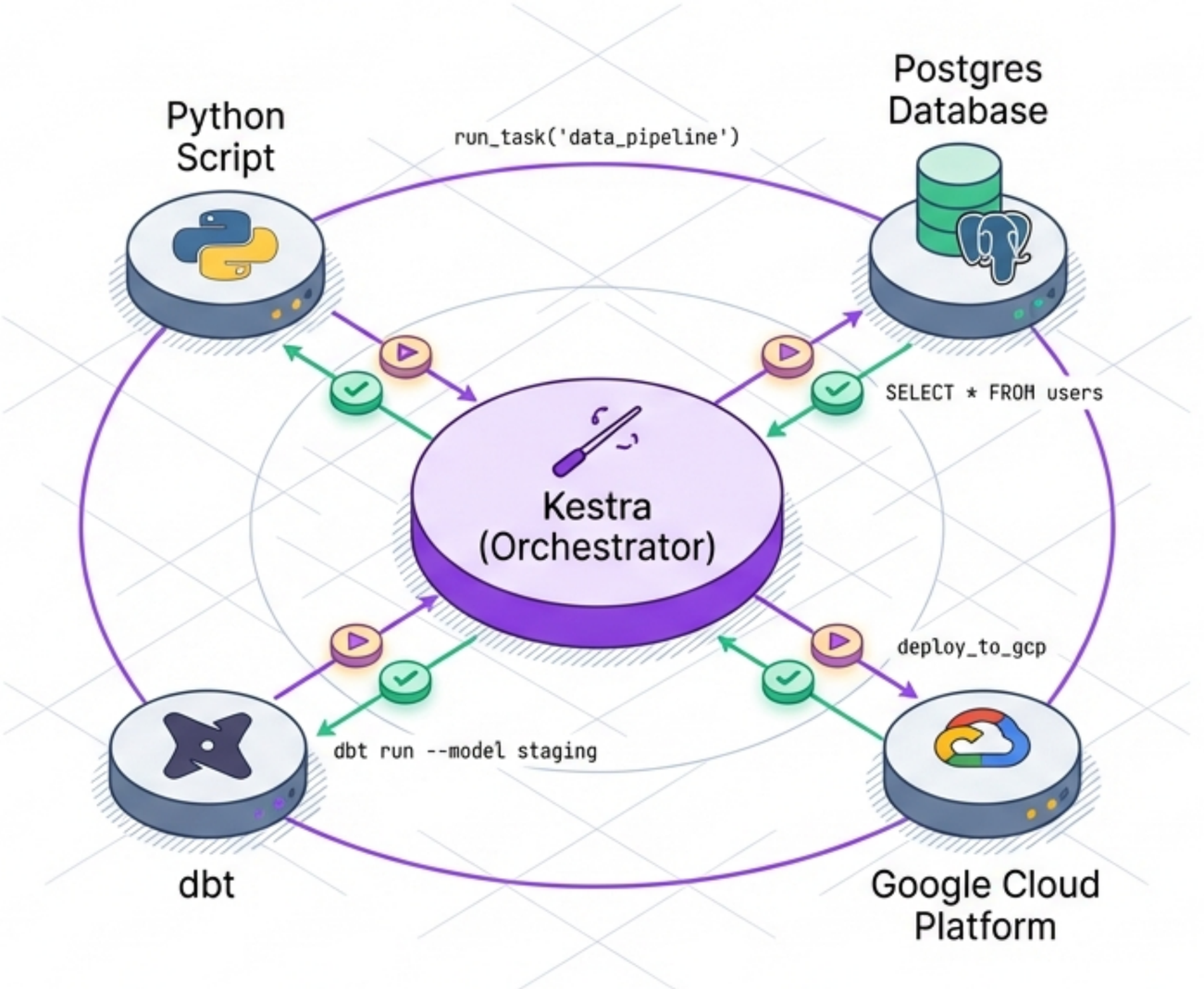
The Conductor and the Orchestra

The Instruments

In a data stack, your tools—Python, Postgres, dbt, DuckDB—are the instruments. Each creates value independently, but without coordination, they cannot create a symphony.

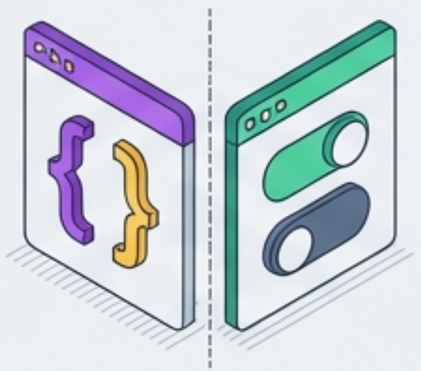
The Conductor

Kestra acts as the orchestrator. It dictates when instruments play, sets the tempo, and **manages** the interactions. It handles logic, error handling, and retries that individual tools cannot manage.



NotebookLM

Why Kestra?
A Unified Platform for Engineers



Versatile Authoring

Build workflows interchangeably using the Code editor (YAML), the No-Code UI topology view, or the built-in AI Co-pilot to generate syntax.

Language Agnostic

While Python is standard, Kestra is not limited to it. Run high-performance tasks in Go, Rust, Java, or C whenever necessary.

Infinite Scale

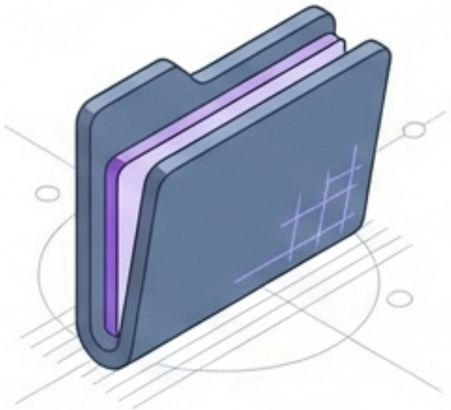
Choose your runner. Execute tasks via Docker for isolated dependencies or Process for speed on the local machine.

Features over 1,000+ dedicated plugins, connecting everything from GitHub to BigQuery out of the box.

NotebookLM

The Grammar of a Workflow

Visual Dictionary

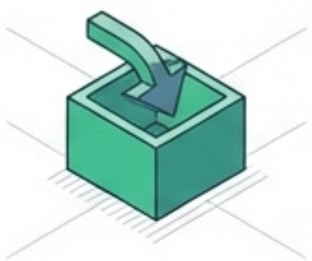


Flow

The container for the pipeline. Defined by a ‘Namespace’ and an ‘ID’ . Once saved, these are immutable.

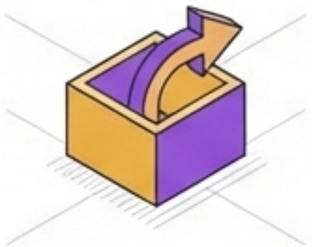
Tasks

The steps within the flow (e.g., Log, Sleep, Python Script). Each has a ‘Type’ and ‘Properties’.



JetBrains Mono Inputs

Dynamic parameters passed at execution start (e.g., selecting a Date or Dataset).



Outputs

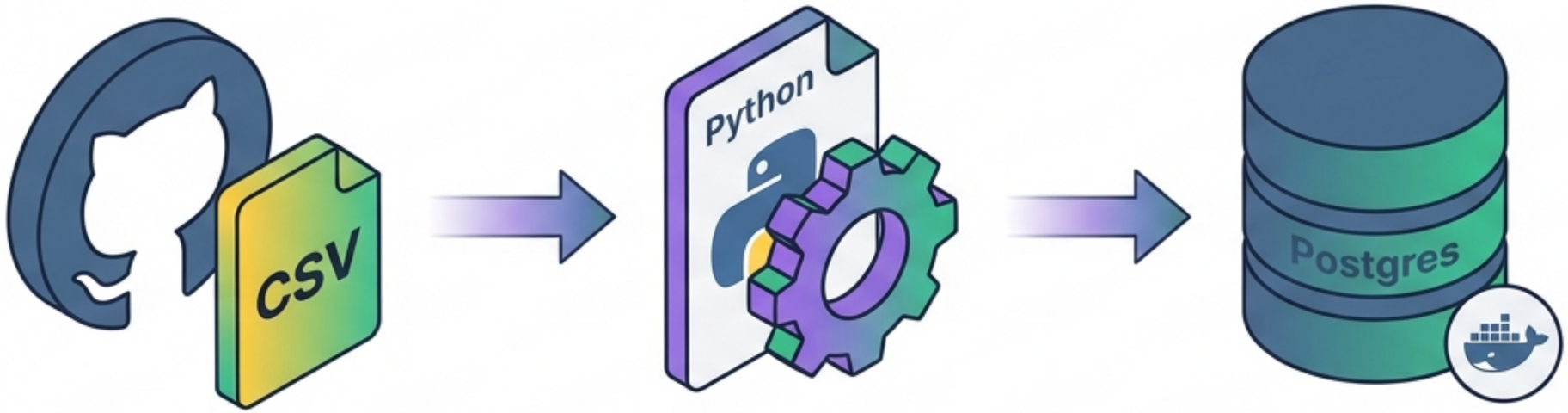
Data generated by a task (e.g., a downloaded file path) accessible to downstream tasks.

Triggers

Mechanisms that automatically start a flow based on a Schedule (Time) or Event (Data arrival).

NotebookLM

The Mission: Local ETL with Postgres



Extract

Download CSV (Yellow/Green Taxi) via Shell

Inter

Transform

Decompress & Add Metadata (IDs).

Inter

Load

Insert into Local Postgres (Docker).

Inter

The Challenge: Variable Inputs

Taxi Color (Yellow vs. Green)

JetBrains Mono

Yellow

Green

Year (2019, 2020)

JetBrains Mono

2019

2020

Month (01-12)

01-12

Inter

NotebookLM

Handling Logic and State

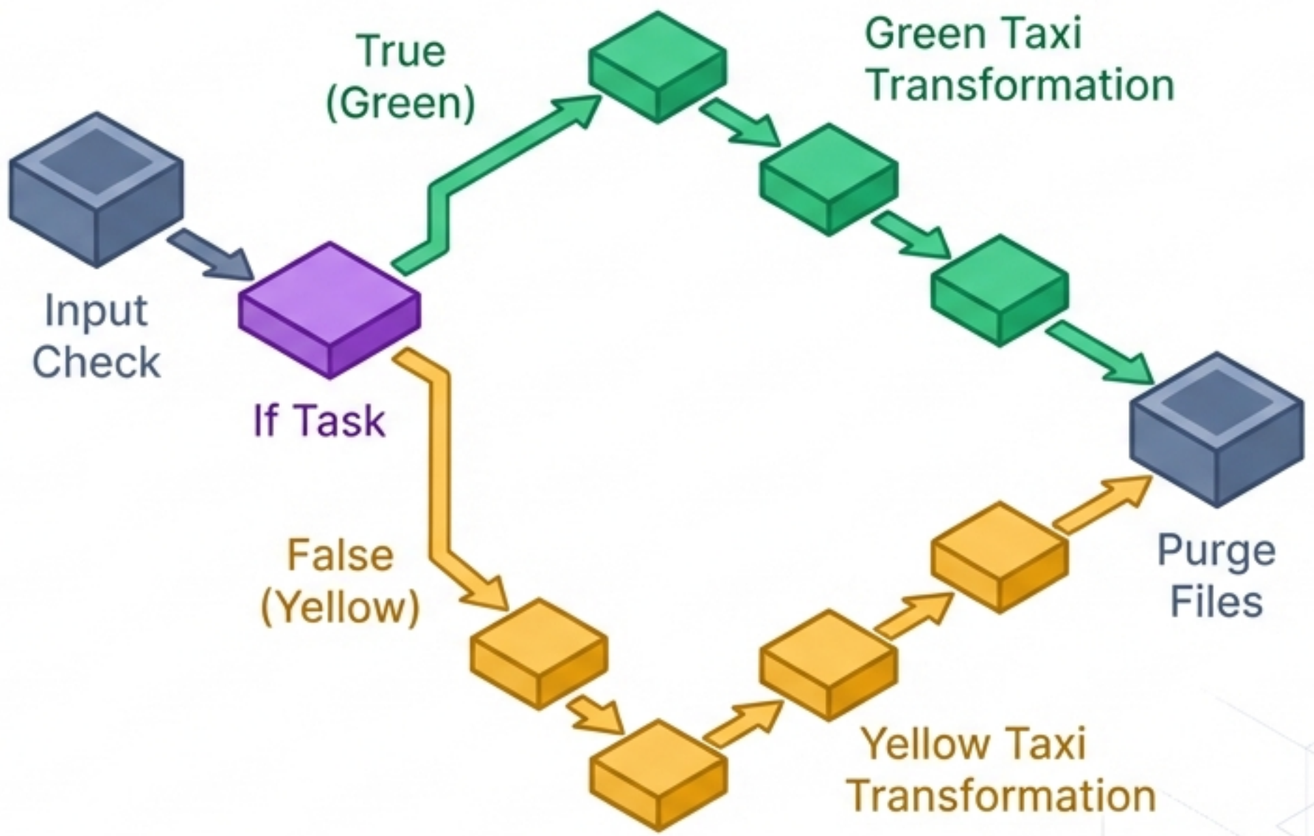
The Problem

The “Yellow” and “Green” taxi CSVs have different schemas. A linear pipeline fails when mapping mismatched columns.

The Solution

The “If” Task.

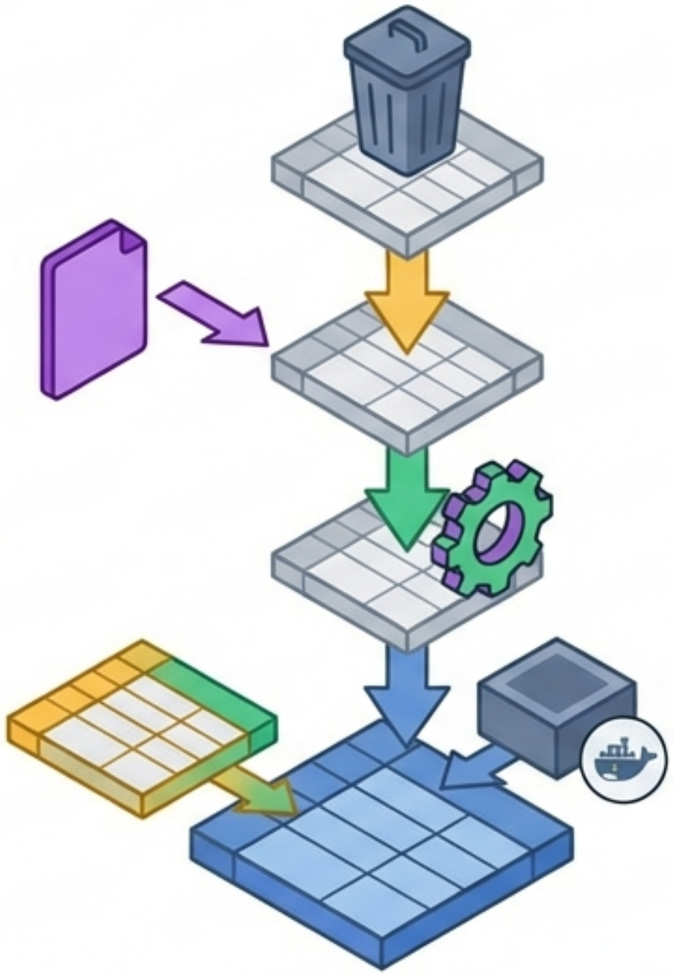
Logic
checks `inputs.taxi == 'green'` to decide execution branch.



NotebookLM

Data Integrity: The Staging Strategy

Ensuring Idempotency with Merge



Truncate Staging

Clear temporary table.

Copy In

Columns like unique_row_id are NULL.

Transform (SQL)

Generate unique_row_id via MD5 Hash.

Merge

Insert/Update based on unique_row_id.

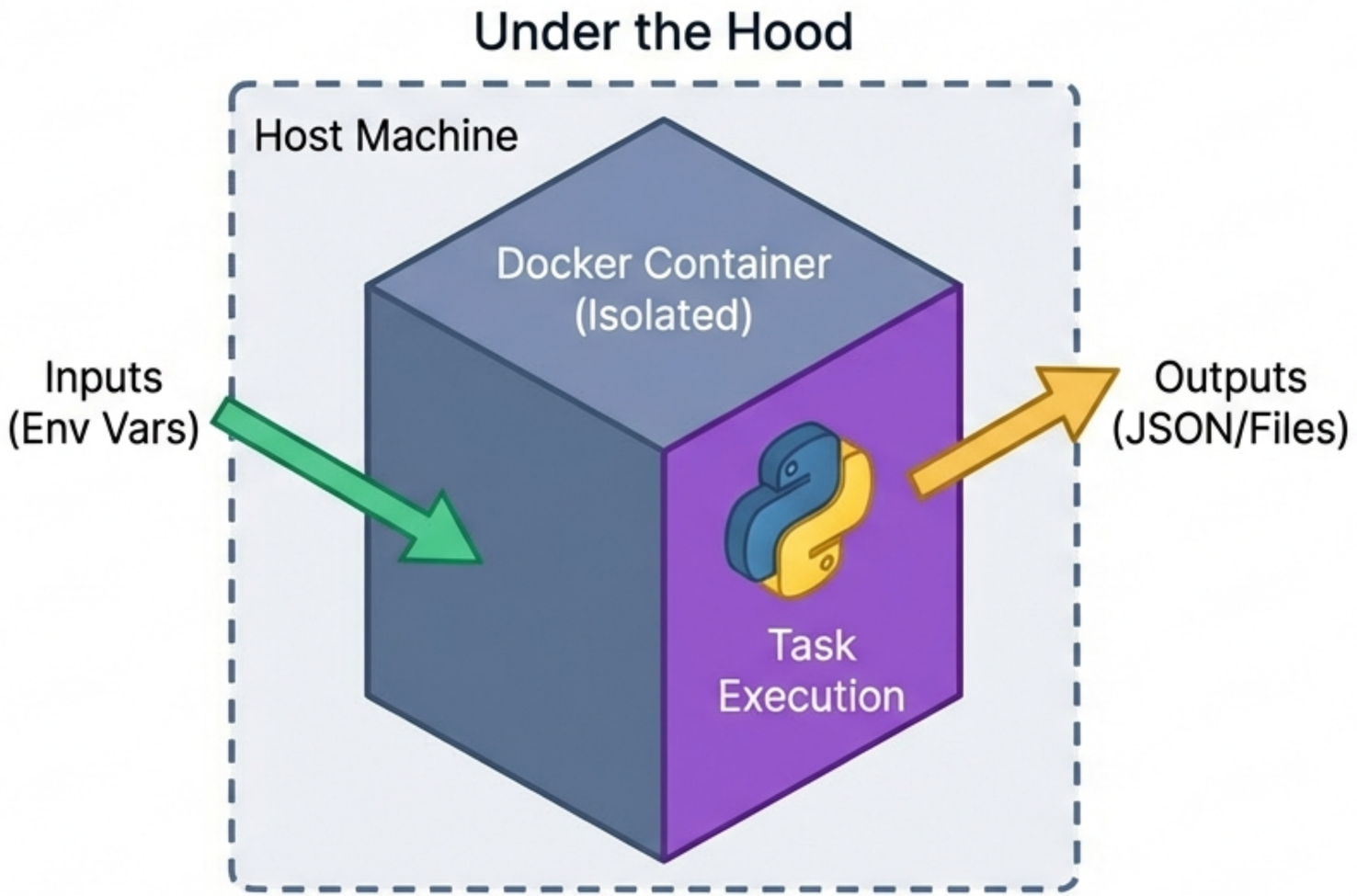
```
MERGE INTO final_table USING staging_table
ON (final.id = staging.id)...
```

Why?

This allows re-running pipelines for the same month without creating duplicate records.

NotebookLM

Orchestrating Python



Execution Model

Kestra spins up a Docker container, installs dependencies (using `uv`), runs code, and destroys the container. Ensures zero dependency conflicts.

Data Passing

Use `from kestra import Kestra` and `Kestra.outputs({'key': 'value'})` to pass metrics.

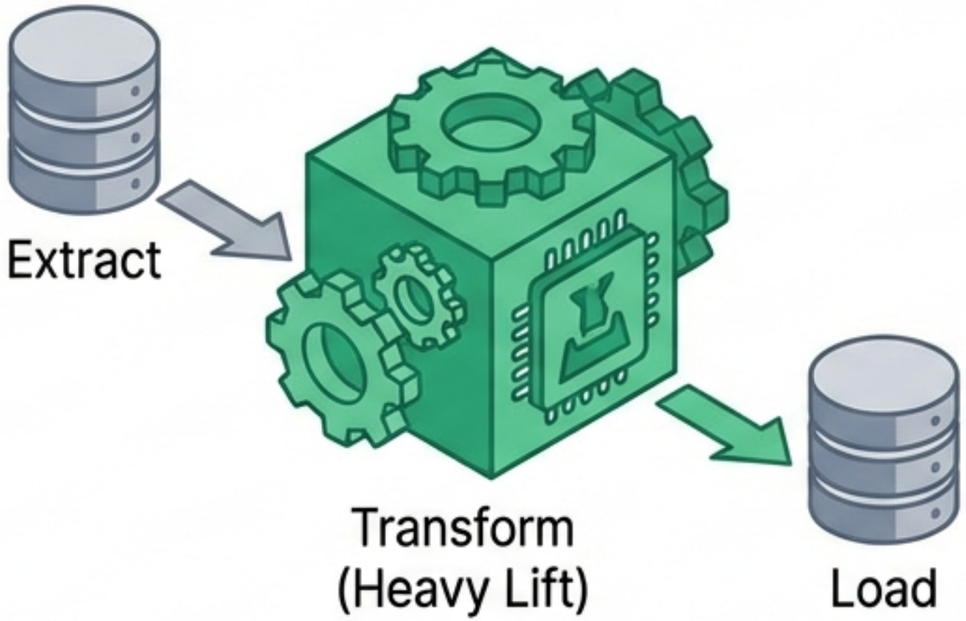
Script Types

Inline Script (YAML snippets) vs. Commands (Namespace Files).

NotebookLM

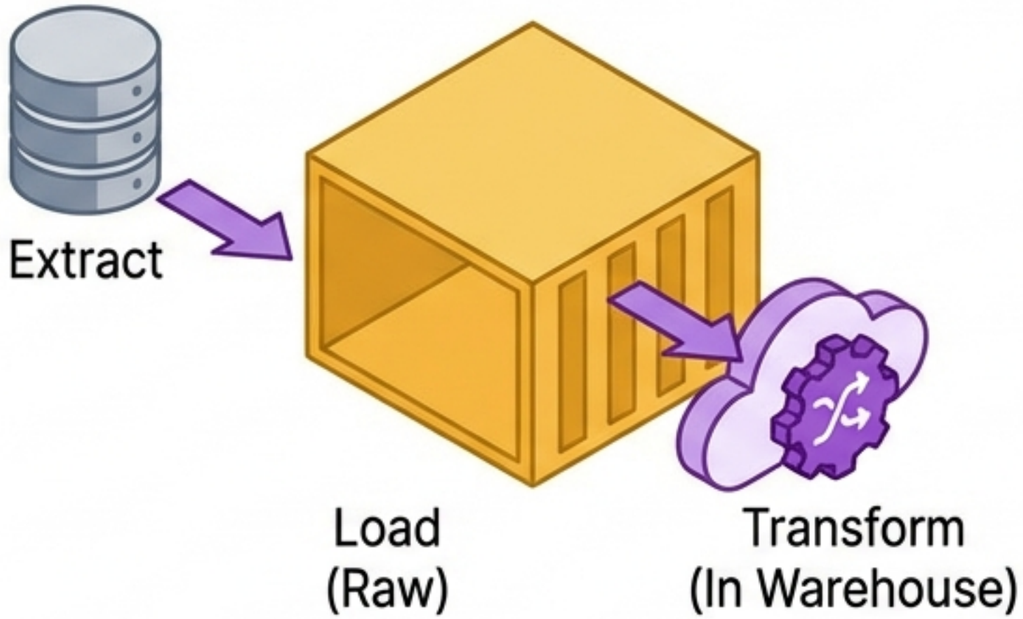
The Pivot: From ETL to ELT

ETL (Local Postgres)



Bottleneck: Local resources strained by processing millions of rows.

ELT (Cloud BigQuery)

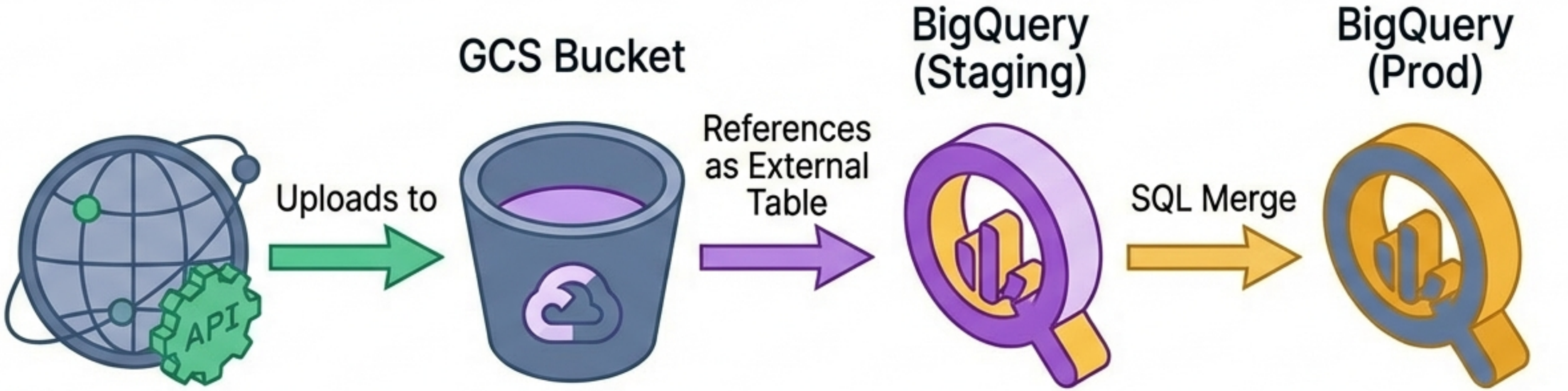


Benefit: Leverages distributed cloud computing for heavy transformations.

Moving transformation to the destination (BigQuery) unlocks the power of the cloud.

NotebookLM

The Google Cloud Architecture



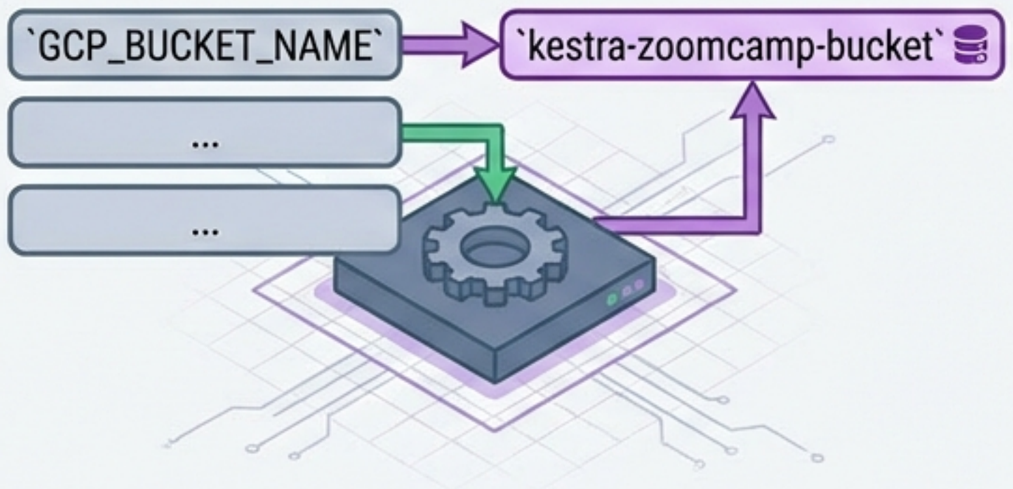
Naming Hierarchy:
ProjectID . Dataset . Table

NotebookLM

Managing Configuration & Secrets

⚠️ Avoid hardcoding sensitive values like Service Accounts or Project IDs.

KV Store (Global Variables)




```
{{ kv('GCP_BUCKET_NAME') }}
```

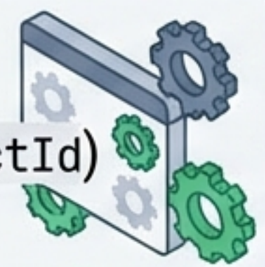
Secrets (Encrypted)

GCP Service Account JSON.

Secure connection

Plugin Defaults

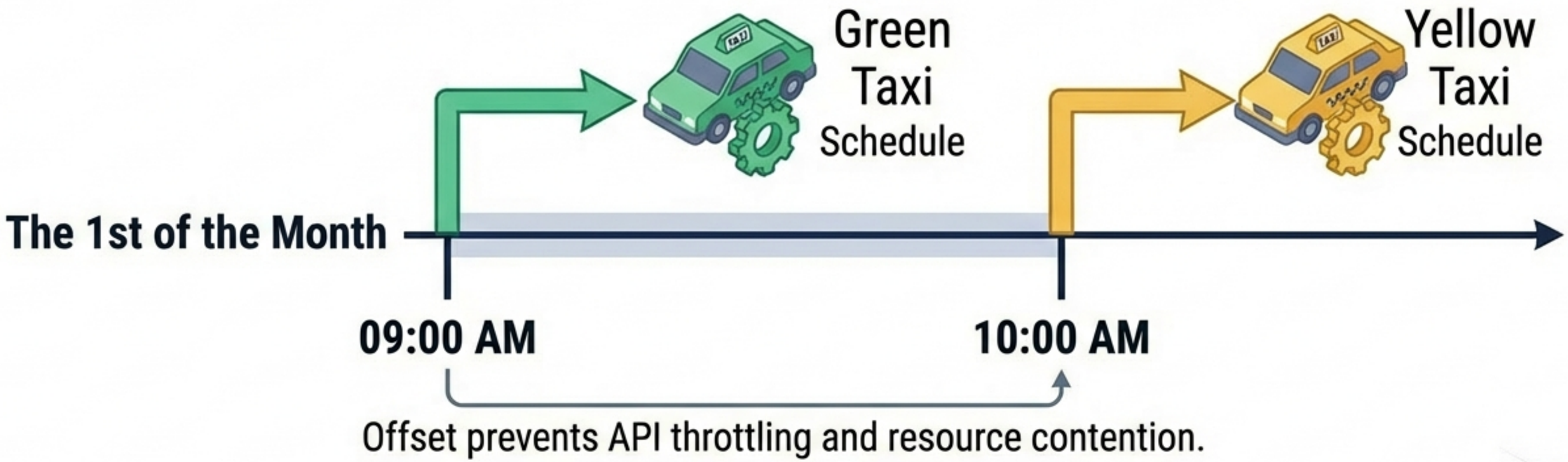
Define properties once (e.g., `projectId`) to apply to all tasks automatically.



NotebookLM

Automating Time: Schedules & Triggers

Decoupling Execution from Manual Input

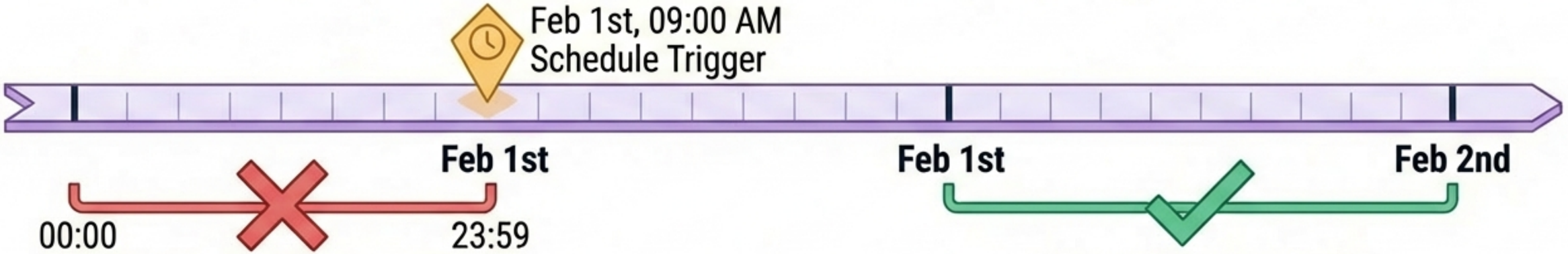


Replaced Input Variables: ~~`{{ inputs.date }}`~~ → `{{ trigger.date }}`

NotebookLM

Backfilling History: The Date Range Nuance

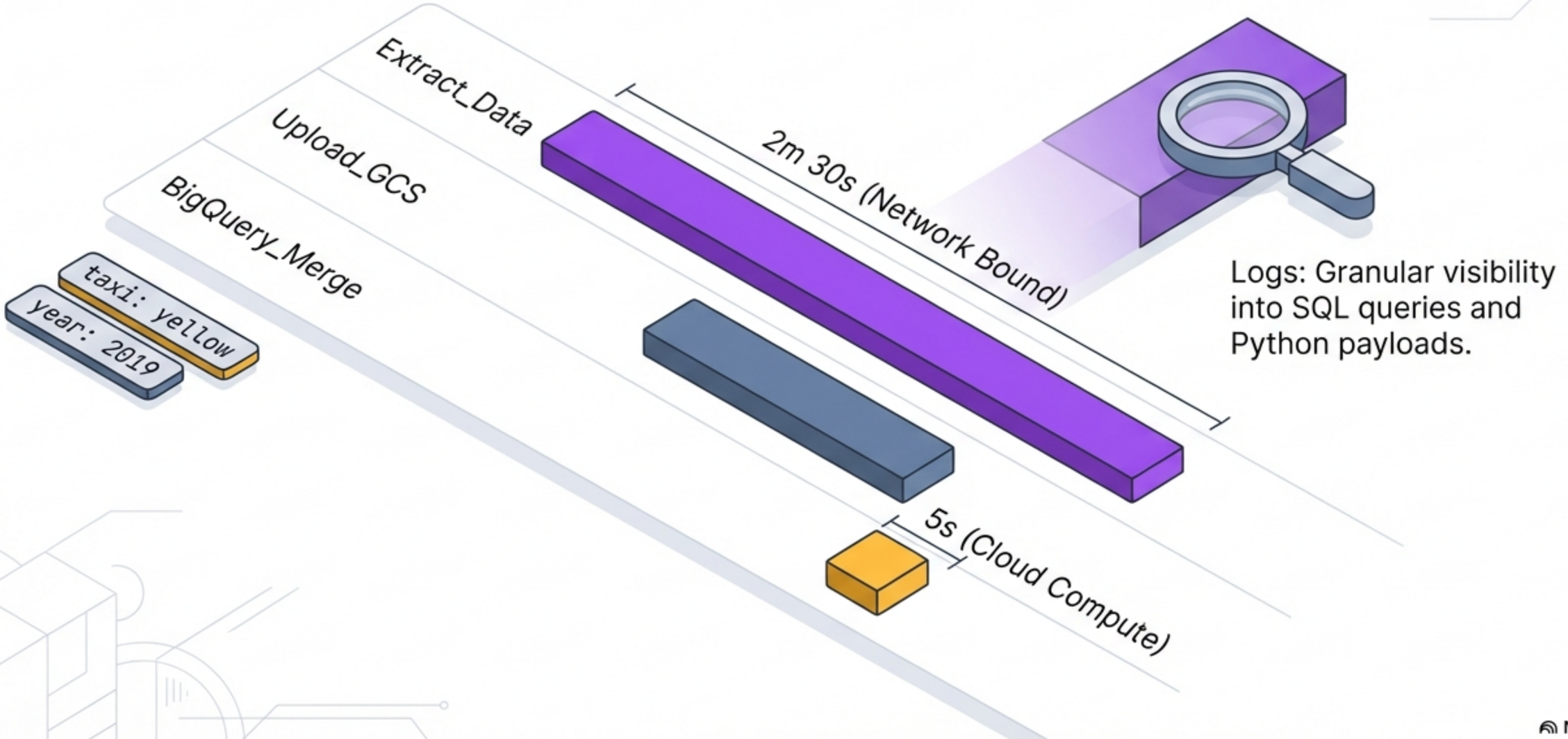
Backfilling allows re-running pipelines for past dates. The execution window must cover the specific trigger time.



Kestra generates executions for every month in the valid range.

NotebookLM

Observability & Visualizing Execution



NotebookLM

The Complete Journey

Orchestrated

Managed Dependencies

Flexible

Handled Schema Changes

Automated

Backfilled History

Scalable

Scaled to ELT

The modern data engineer doesn’t just write scripts; they build resilient, observable, and automated ecosystems.

NotebookLM