Data Sync Pro

#Trigger: Automation & Validation

Replace Apex Triggers and Record-Triggered Flows with simple, scalable, maintainable records — rules-based, bulk-safe, and built to drive technical debt toward zero.

Data Sync Pro

The Salesforce Automation Challenges

1

Logic Goes Deep and Sprawls

In Apex, logic nests; in Flow, canvases sprawl. Either way, complexity compounds — until small changes carry big risk.

2

Bulk & Performance Traps

Apex needs hand-crafted bulkification; Flows hit performance walls at scale — demanding deep platform expertise that takes years to master.

3

Coupled, Not Contained

Apex classes call classes. Flows invoke subflows. The chains run many levels deep — and over time, one small change ripples through many.

4

Scattered, Not Governed

With Apex and Flow, the rules that drive one process are split across many files — no single view to read, audit, or change them as a whole.

5

Code Complexity Snowballs

Code costs more to build, read, and change than a declared rule. As Apex and Flow accumulate, that cost compounds — every change inherits the weight of everything before it.

6

Tech Debt Compounds

As Apex and Flow logic accumulates, tech debt builds — quietly, steadily. Modularity isn't built in — it's a discipline. And without enforcement, even careful teams watch boundaries blur over time.

From Sprawl to a Single Record

BEFORE · Logic scattered across code and canvas

To automate one Account object today, you build all of:

APEX	AccountTrigger.cls · AccountHandler.cls · RecursionGuard.cls · AccountHandlerTest.cls
FLOWS	Account_BeforeUpdate · Account_AfterInsert
SUBFLOWS	NotifyOwner · ValidateFields

…coordinated by hand. Every change touches multiple files.

AFTER · One #Trigger Executable

Executable: Account Rollups

#Trigger · Object: Account

DML EVENT
before update

SCOPING
Industry = "Technology"

MATCH
Self (the Account)

MAPPING

CaseCount__c	← AGG_COUNT(Case)
OpenOppAmount__c	← AGG_SUM(Opportunity)
LargestOppAmount__c	← AGG_MAX(Opportunity)

ACTION
Update fields on record

Modular Triggers. Zero Tech Debt.

1

Flat & Rules-Driven

Automation and validation rules live as Executable records — never nested, never sprawling. Every rule stays flat, readable, and straightforward to change.

2

Bulk-Safe Engine

Write declarative rules as if for one record — the engine automatically bulkifies queries, aggregations, and DMLs across all Executables as one optimized plan.

3

Self-Contained Executables

Each Executable is self-contained — no cross-class chains, no subflow webs, no hidden dependencies.

4

Grouped & Governed

Group #Trigger Executables into Pipelines however makes sense — one per object, or multiple with segregated scope filters. Manage them like any related list: navigate, filter, and reorder in one place.

5

Rules, Not Code

Each Executable is a regular Salesforce record, created and updated through a guided, step-by-step interface. All logic is bounded within that record — never sprawling, and far simpler to change than code.

6

Modularity Enforced

Modularity is structural, not a discipline. Every Executable is a self-contained, bounded module — a record with clear input, transformation logic, and output. Debt stops compounding.

#Trigger Features

Replace Apex triggers and Record-Triggered Flows with declarative, rules-driven records — simple, bulk-safe, and modular by design.

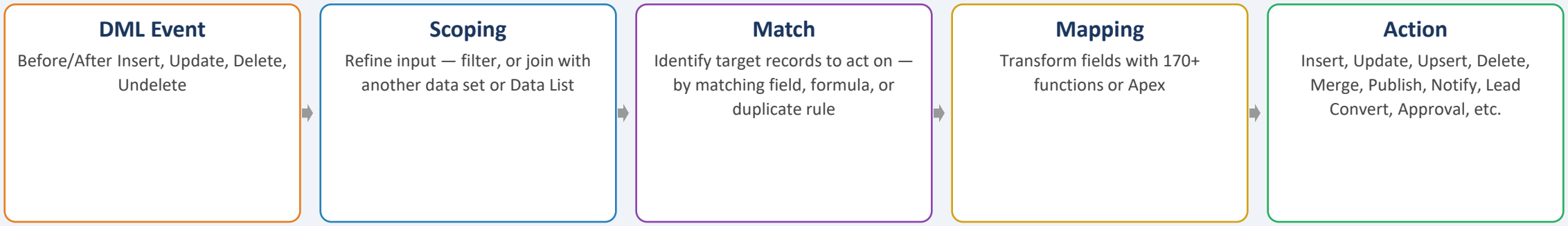
What Admins Build, Not Developers

- **Records, Not Code** — Every rule is a Salesforce record. Admins build, edit, and audit it through a guided UI — no Apex, no canvas, no deploys.
- **Modularity Enforced** — Each Executable is bounded: one event, one scope, one transformation, one action. Boundaries are guaranteed by the engine, not by discipline.
- **Bulk-Safe Out of the Box** — Functions, parent traversals, and AGG_ rollups bulkify automatically across all rules on an object. No governor traps, no manual collection management.
- **Manage at Scale** — Designers group Executables into Pipelines that fit their domain, with each Pipeline organizing its rules by event for ordered, filtered review. No hunting through files or canvases.

Key Features

- **All DML Event Hooks**
- **Self-Adaptive Trigger**
- **Cross-Record Trigger Actions**
- **Per-Event Scope Filters**
- **Recursion Prevention Control**
- **Cross-Trigger Variables**
- **Per-Executable Record-Based Sharing**
- **Per-Executable Bypass & Access Permissions**
- **Cross-Org Target Action**
- **170+ Built-In Functions, Extensible via Apex**

DML Event → Five Declarative Stages — Event, Scoping, Match, Mapping, Action — all in one record.



Why #Trigger

100%

Bulk-Safe

Built for bulk, out of the box

10×

Faster Changes

Edit a record, not a class graph or canvas web

0

Code Skills Required

Admins configure entire UIs — no developer dependency

→ 0

Tech Debt

Drive tech debt toward zero with declarative, modular rules

Safe by Default

Every rule runs governor-aware from single records to high-volume batches. No SOQL-in-loop traps, no per-record boilerplate, no rewrites at scale.

Accelerate Delivery

Update a rule in minutes — open the record, change the field, save. No deploys, no test refactors, no canvas rewiring.

Admin-Owned

Build, change, and govern automation and validation rules in a Setup-style interface. Admins own the full lifecycle — activation, conditions, and updates included.

Modularity Enforced

Every Executable is a bounded record with clear input, logic, and output. Tech debt stops compounding — by design, not discipline.