The Ultimate Guide in choosing Salesforce Triggers or Flows

Kalpana Puli[1], Sai Rakesh Puli[2]

[1, 2]Independent Researcher, Texas, USA
[1]kalpanapuli@gmail.com, [2]Sairakesh.puli@salesforce.com

Abstract

Perhaps the hardest decision a Salesforce Specialist needs to make is choosing between enabling automation via a trigger or through a flow. This choice has significant implications for project success, maintainability, and scalability of a Salesforce implementation over time. Success depends on a firm understanding of each tool's capabilities and limitations.

We will cover key elements of both Flow and Trigger, which will help in choosing the right approach as part of your project implementation. Everything about Salesforce automation from Apex triggers to different flow types, including ground implementation examples will be included. The content will present practical frameworks to help you make smart decisions based on your business needs, data volumes, and complexity levels.

Keywords: Salesforce Automation, Apex Triggers, Flow Builder, Workflow Rules, Process Builder, Record-Triggered Flows, Screen Flows, Scheduled Flows, Platform Event-Triggered Flows, Salesforce Flows, Before-Save Updates, After-Save Actions, Automation Tools, Salesforce Automation Evolution, Trigger Handler Pattern, Context Variables, Business Logic Complexity, Data Volume Considerations, Debugging Salesforce, Scalability, Performance Optimization, Error Handling, High-Volume Data Processing, Transaction Control, External Integration, Declarative Tools, Programmatic Tools, Salesforce Automation Comparison, Decision Framework.

The Evolution of Salesforce Automation

Salesforce automation has revolutionized dramatically since it first began. The understanding of this development provides significant insights to make informed decisions about automation tools today.

From Workflow Rules to Process Builder

Workflow Rules became Salesforce's first automation tool and provided simple yet reliable functionality [1]. These rules mastered four simple functions:

- Field updates
- Task creation
- Email alerts
- Outbound messages

Process Builder emerged as a fresh approach in Winter 2015 with a visual interface that transformed how administrators handled automation [2]. It expanded automation capabilities by a lot while keeping a user-friendly approach. Process Builder marked a significant step forward that allowed administrators to build multi-decision processes and manage complex business requirements [3].

Introduction of Salesforce Flows

Flow Builder stands out as Salesforce's most advanced declarative automation tool. The platform started its journey as Cloud Flow Designer in 2012 and went through a complete transformation to become the modern Flow Builder platform in 2019 [2]. Flow's capabilities are way beyond the reach and influence of its predecessors. The platform provides:

1. Screen flows for guided user experiences
2. Record-triggered flows for automated processes
3. Scheduled flows for time-based automation
4. Platform event-triggered flows for real-time responses

Salesforce announced a major change in their automation strategy at Dreamforce '21. The company positioned Flow as the future of declarative automation on the platform [4]. This strategic move shows Salesforce's steadfast dedication to unifying automation tools and boosting the platform's capabilities.

The Role of Apex Triggers

As Salesforce automation continues to advance, Apex triggers remain the programmatic backbone of the platform. They give developers exact control over database events and enable complex operations beyond declarative tools' capabilities [5]. Apex triggers are still the preferred solution for:

- Complex data validation
- Multi-object updates
- High-volume data processing
- External system integration

Declarative tools and Apex triggers now work together in a complementary partnership. Each handles specific use cases based on complexity and scale [6]. This development has created a complete automation ecosystem that serves both administrators and developers, offering multiple ways to reach business goals.

Deep Dive into Salesforce Flows

Salesforce Flow serves as the lifeblood of modern automation that brings developers and administrators together with its versatile capabilities [7]. The three main types of flows drive Salesforce automation solutions effectively.

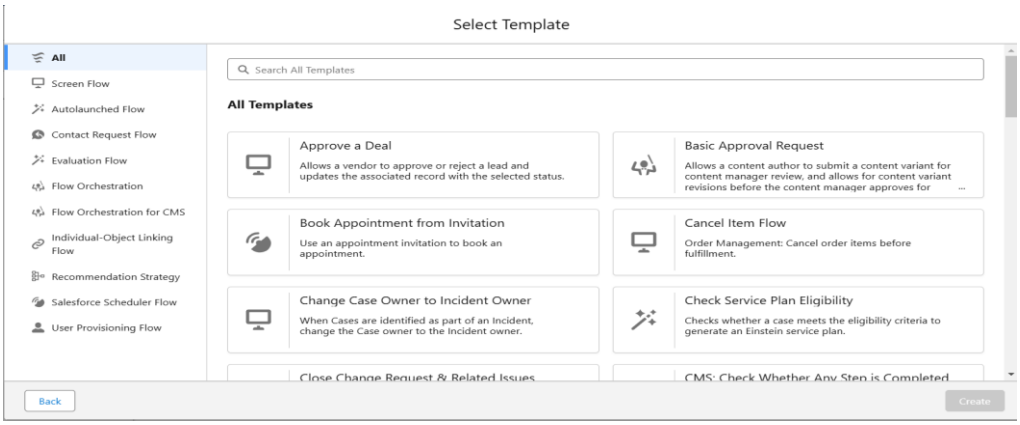

Fig I Salesforce Flows [1]

Record-Triggered Flows

Record-triggered flows serve as the backbone of automated processes. The system automatically activates these flows as records get created, updated, or deleted [8]. Background operations run smoothly without any

user interaction. These flows process up to 50,000 records within governor limits [9]. This makes them ideal for most business scenarios.

Key capabilities include:

- Before-save updates run 10 times faster than traditional processes [8]
- After-save actions handle complex multi-object operations
- Conditional logic responds to record changes
- Automated email alerts and notifications streamline communication

Screen Flows

Screen flows add interactive aspects to automation and create guided experiences for users. Now, users can find their way to these flows via pages in Lightning, quick actions, and the utility bar [10]. The potential of screen flows is that they can break down any complex process into few easy steps. It presents intricate work in the form of some easy steps and derives the input from the users quickly.

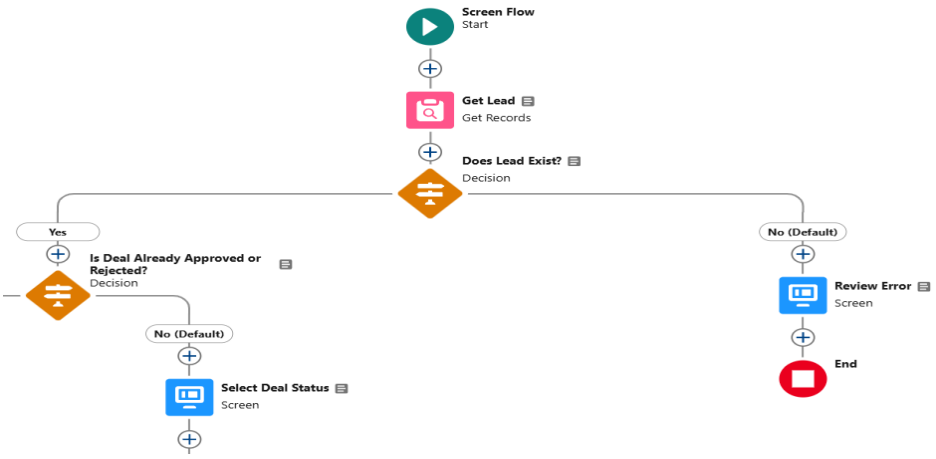

Fig II Screen Flows [2]

Developers must take into consideration component visibility features in
Their implementation. They render or hide elements depending on what input the user provides [10]. This manner, users shall only view the information they need, and, hence, their experience becomes more efficient and error-free.

Scheduled Flows

Scheduled flows help automate routine tasks by running at specific times and frequencies [9]. These flows run in three ways:

1. Once for one-time bulk updates
2. Daily for regular maintenance tasks
3. Weekly for periodic data processing

Scheduled flows can process only 200 records per batch, while batch Apex handles 2,000 records [11]. Organizations can run 250,000 scheduled flow interviews daily, or 200 times their user license count, whichever is greater [9].

Administrators need to follow these key practices to make flows work better:

1. Create clear documentation for future updates

2. Use subflows as reusable components
3. Keep logic flexible instead of hard coding
4. Add bypasses for data loads and sandbox seeding [7]

Flow Builder's accessible interface helps both administrators and developers. Teams should know its limitations well. Flows must stay within Apex governor limits and need careful planning for data volume and complexity [12]. High-volume scenarios or complex integrations might need traditional apex triggers instead of flows, depending on specific needs.

Understanding Apex Triggers

Apex triggers form a rich set of programming languages in the Salesforce automation that enables a custom action to be performed before or after a certain occurrence within the database. Developers can use this to provide intricate business logic. Its performance is excellent while processing large data volumes.

Before and After Triggers

Before vs. After Triggers Difference Basing effective automation solutions requires identifying the differences. Before triggers update or confirm values of records just before saving to a database [14]. The best usage for these triggers is when data needs to be validated or transformed before committing to the database.

Triggers occur after triggers are activated whenever developers need to work with system-generated field values or whenever they must change related records [15]. These triggers are very useful in various scenarios:

Using system-generated fields such as IDs on records.

Generation of or updating related records

Complex cross-object logic

Trigger Context Variables

Developers can learn about the runtime environment through the System. Trigger class. These variables are a great way to get insights into the trigger's execution context and the affected records.

The most used context variables include:

- Trigger.new: Contains new versions of sObject records
- Trigger.old: Holds previous versions of records
- Trigger.newMap: Provides ID-to-record mapping for new versions
- Trigger.oldMap: Maps IDs to previous record versions.

Keep in mind that these variables become available based on specific trigger contexts. To name just one example, Trigger.new works only with insert, update, and undelete triggers, while Trigger.old becomes available in update and delete triggers.

Trigger Handler Pattern

The Trigger Handler pattern stands out as a best practice that helps developers organize and maintain Apex triggers effectively. Developers use this pattern to separate trigger logic into distinct classes that promote code reuse and easier maintenance.

Key benefits make this pattern valuable:

Better organized code
Boosted maintainability
Simplified unit testing

Tighter control over trigger execution order

The pattern's core principle requires one trigger per object. This approach prevents execution order conflicts and creates a more manageable codebase. Three main components make up this pattern: the trigger itself, a handler class that implements logic, and a helper class containing reusable business logic for different services.

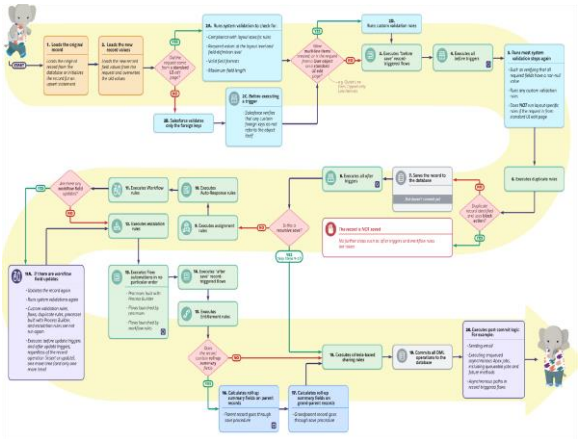

FIG III Trigger Handler Pattern [3]

High data volumes and complex business requirements trigger a superior choice over declarative alternatives. Triggers deliver better performance and precise control. They shine in situations that need:

- Complex business rule validation
- Processing of high-volume data
- Exact execution order control
- Advanced capabilities for exception handling and debugging

Comparing Flows and Triggers

Developers need to understand the practical differences between flows and triggers to implement Salesforce automation tools successfully. These tools differ in several significant dimensions that affect development and maintenance tasks daily. Let's examine how they stack up against each other.

Ease of Implementation

Flow Builder's accessibility makes it an excellent choice for organizations with teams of varying technical expertise. Teams can create and modify workflows through a visual interface without tucking into complex code. The visual elements help teams grasp processes better and make changes with ease.

But Apex triggers give teams more customization options and flexibility when dealing with complex scenarios. These triggers need specialized programming skills and more maintenance work. They provide exact control over automation logic that flows cannot match.

Debugging and Troubleshooting

Flow and Apex trigger each handle debugging and error management differently.

Flow Debugging Features:

- A visual debug button sits right on the Flow Builder canvas
- Runtime options adapt to different flow types
- Error notifications arrive through email
- Users can check failed flow versions easily

Apex triggers shine with their testing capabilities through unit tests. Test coverage requirements help maintain code quality and catch problems early in deployment. These triggers also come with advanced error handling and logging features that work great when detailed error tracking matters.

Scalability and Performance

The performance gap between flows and triggers becomes clear when you deal with high volumes. These metrics tell the story:

- Complex operations in flows need three times more CPU time than Apex triggers
- Before-save flows run 10-20x faster than workflow rules for bulk same-record updates
- Single-record operations in Apex triggers use almost no CPU time while flows need ~100ms

Triggers shine at handling large data volumes. Their superior performance comes from three main advantages:

- They know how to use static methods to cache data
- They control transaction scope better
- They handle bulk operations more smoothly

Performance Considerations: Complex implementations make this performance gap even wider. Teams might see processing delays of several seconds on their account and opportunity objects when multiple processes stack up. These delays matter most:

- During bulk data updates
- With complex business logic
- In multiple concurrent operations
- For time-sensitive transactions

The before-save flows are nowhere near as slow as traditional workflow rules and Process Builder. Teams without many developers find flows a great way to get decent performance while keeping things simple.

Your choice between triggers and flows depends on what your organization needs. Triggers excel at speed-critical tasks. Flows need less technical knowledge to maintain and work well enough for many business processes. Teams should look at their technical skills, automation complexity, and speed requirements to make the right choice.

Decision Framework for Choosing Between Flows and Triggers

Teams need to think about multiple factors to make smart decisions about automation tools. A well-laid-out approach can help teams effectively choose between flows and triggers.

Complexity of Business Logic

	Low Code	---------->		Pro Code
	Before-Save Flow Trigger	After-Save Flow Trigger	After-Save Flow Trigger + Apex	Apex Triggers
Same-Record Field Updates	Available	Not Ideal	Not Ideal	Available
High-Performance Batch Processing	Not Ideal	Not Ideal	Not Ideal	Available
Cross-Object CRUD	Not Available	Available	Available	Available
Asynchronous Processing	Not Available	Available	Available	Available
Complex List Processing	Not Available	Not Ideal	Available	Available
Custom Validation Errors	Not Available	Not Available	Not Available	Available

FIG IV business Logic [4]

Business requirements' complexity is a vital factor in selecting the right tool. Flows work best with simple automation tasks but face challenges with extensive data processing and transformation. Teams need to think about several key factors:

- Data transformation requirements
- Integration with external systems
- Transaction control needs
- Error handling sophistication
- Bulk processing requirements

Complex list processing tasks like in-place data transforms, sorts, and filters perform better in Apex than flows. Triggers offer enhanced control and flexibility at the time teams need precise computation management or database action control.

Data Volume Considerations

Data volume substantially affects automation performance and reliability. Here's a quickest way to make volume-based decisions:

Scenario	Recommended Tool	Reason
Basic record updates (<1000)	Flow	Easier to implement and maintain
Large data volumes	Trigger	Better handling of bulk operations
Complex formulas	Trigger	More efficient processing
Same-record updates	Before-save Flow	10-20x faster than traditional tools

Fig V Data Volume [5]

Your organization should assume every automation will handle large data volumes without notice. This approach will give adaptable solutions as business needs grow. Teams should remember that flows have specific limitations. They can process only up to 250,000 records per day for scheduled operations.

Maintenance and Future Extensibility

Sustainable automation solutions need adaptable maintenance and scalability plans. Teams with limited developer resources will find flow triggers a compelling option. These are more performant and easier to debug, maintain, and extend than previous no-code offerings.

Key maintenance factors include:

- Team composition and skill sets
- Existing automation frameworks
- CI/CD pipeline requirements
- Code quality management practices

Teams should think about how flows can become challenging to manage as new requirements emerge. Organizations with well-laid-out CI/CD pipelines and trigger frameworks might find it more affordable to stick with trigger-based solutions.

A hybrid approach works best for mixed-skill teams. This approach includes:

- Using flows for straightforward business processes
- Implementing triggers for complex computations
- Leveraging invocable Apex within flows for specific functionality
- Maintaining clear documentation for both approaches

Note that flows offer quick implementation but may hit limitations as projects grow more complex. Triggers provide improved control and flexibility but need specialized programming skills and greater maintenance efforts.

Real-World Examples and Case Studies

Real-life examples show how choosing between flows and triggers affects Salesforce automation results. Organizations have used these tools successfully to meet their business goals.

Simple Automation with Flows

Organizations achieve great success when they use flows for basic automation needs. OpenTable's flow-based automation has substantially improved its customer service operations through AI-powered service agents. Saks has raised its luxury shopping experience by using flows that create unified data processes.

RBC Wealth Management stands out as a powerful example. The company revolutionized its client onboarding process with flow automation and cut down account opening time from 4 days to just 24 minutes. This dramatic improvement shows how flows excel at streamlining customer-facing processes.

Success stories include:

- Vonage cut response times from 4 days to 4 minutes with flow automation
- Texas Tech unified 8 different systems into one comprehensive view using flows.
- Decathlon delivered products to market 50% faster through automated analytics

Complex Data Processing with Triggers

Organizations find triggers indispensable for high-volume data processing tasks. United Tractors demonstrates this through their implementation of trigger-based automation to manage field service operations. The company's mobile workforce efficiency and visibility improved significantly.

Triggers show their performance advantages in these scenarios:

- Processing up to 250,000 records per day for scheduled operations
- Handling complex validation errors with custom messaging
- Managing high-performance batch processing with transaction safety points.

	Record-Changed Flow: Before Save	Record-Changed Flow: After Save	Record-Changed Flow: After Save + Apex	Apex Triggers
High-Performance Batch Processing	Not Ideal	Not Ideal	Not Ideal	Available

Fig VI Performance Metrics Table [5]

Hybrid Solutions

Companies found there was a sweet spot in combining flows and triggers. ICT brings both methods together. They provide fast and seamless services using AI-based activities. The hybrid architecture they have enables them to do the following:

- Serve complex data processing by using triggers
- Keep the interface user-friendly by making use of flows
- Perform better in various applications

Lion Parcel proves how effective this hybrid method can be. With a combination of both automation approaches, they managed to achieve cost efficiency that was 40% better.The Southeast Asian fintech, Xendit, achieved the same strategy and increased the productivity by 25%.

Hybrid implementations work because teams know the strengths of every tool. PRISM+ cut their booking time by 95%. They achieve this by implementing Salesforce Field Service, which combines flow for user interaction with triggers for backend work. Companies can utilize the best features of both worlds - flows handle user processes while triggers control complex backend tasks.

Sonak Group has another interesting story. They integrated Einstein 1 with standard automation tools, and their revenue increased 70% year in year out for three consecutive years. Their success draws out that a combination of different automation tools the right way leads to excellent output.

The teams need to consider the following aspects before they select a hybrid solution: -

Needs in terms of data volume
Interface requirements
Complexity in processing
Maintenance Capacity

This is proof that in every case, the right tool picks matter. Flows shine in user processes and simple automations. These are better suited for tasks that deal with complex data processing and high-volume data-related activities. Smart companies use both tools well and build strong solutions that fit their business needs perfectly.

Conclusion

Salesforce Flows and Apex Triggers are two different powerful automation tools that serve others. Teams lacking in deep technical expertise find Flows ideal for user-friendly implementations and simple business processes. Apex Triggers serve data operation complexities and high-volume scenarios with precise control and performance capabilities. Many organizations also find these tools create powerful solutions together to meet their business requirements and keep things simple where it is needed.

Choose Flows or Triggers based on what your team can handle and the longer-term maintenance requirement. Of course, either one yields very powerful automation abilities - the key is fitting the appropriate tool to each use case. The easier a Flow is to build may be easier to implement in a simple context, and once complexity hits, adding the Trigger components is organic.

References

[1] SalesforceBen, "The history and future of Salesforce automation.," SalesforceBen.com, [Online]. Available: https://www.salesforceben.com/the-history-future-of-salesforce-automation/. [Accessed 21April 2023].

[2] Inardua, "An introduction to automation in Salesforce." Inardua.co. https://www.inardua.co/blog/an-introduction-to-automation-in-salesforce [Accessed 23 May 2023].

[3] Simplilearn, "Salesforce automation tools." Simplilearn.com. https://www.simplilearn.com/salesforce-automation-tools-article [Accessed 24June 2023].

[4] SalesforceBen, "Introduction to Salesforce Flow." SalesforceBen.com. https://www.salesforceben.com/introduction-salesforce-flow/ [Accessed 24July 2023].

[5] Salesforce, "Introduction to Apex triggers." Trailhead.Salesforce.com. https://trailhead.salesforce.com/content/learn/modules/apextriggers/apextriggers_intro [Accessed 25Aug 2023].