LaunchDarkly

How Product and Engineering Teams Build Better Features with Trusted Data

Table of Contents

1

Turn your releases into learning opportunities

Shipping software isn't just about writing code and pushing it live. It's about creating better user experiences and business impact with every release. Every new feature release comes with a question:

Will this make things better for our users, or are we about to introduce a problem we didn't see coming?

Product and engineering teams have long relied on instinct, customer feedback, and best practices to guide these decisions. But gut feelings aren't data; even the most experienced teams can get it wrong.

Experimentation changes this. Instead of guessing how a release will perform, teams can test, measure, and know with experimentation. By running controlled experiments, they can see how a change impacts user behavior, business metrics, and system performance before committing to a full rollout.

Experimentation turns a release cycle into a continuous learning cycle.

What experimentation is— and what it's not

At its core, experimentation involves:

- ✓ Randomly assigning users to different variations (A/B testing).
- ✓ Keep assignments stable so that results are easy to interpret.
- ✓ Measuring outcomes with statistical rigor to ensure results are trustworthy.
- ✓ Drawing data-driven conclusions to inform future product decisions.

With these principles, teams don't just react to data—they make informed, confident decisions. They can answer questions like:

- ? Does this feature improve engagement, or is it just adding friction?
- ? Will this checkout flow increase conversions, or do users abandon more often?
- ? Are these AI-driven recommendations helping customers, or are they just noise?

Rather than rolling out features and hoping they work, experimentation proves what works and what doesn't, before changes are permanent. It can reveal what's really happening when you introduce a change; not just correlation, but causation.

Feature flags and percentage rollouts: risk reduction, not experimentation

Feature flags are a must for modern software development. Turning on features for specific targeted audiences rather than “all-or-nothing” releases significantly reduces risk.

---

200 milliseconds or less

— that’s all it takes to turn off a broken feature and avoid complicated rollback plans and all-night redeploys.

They let teams gradually release new functionality, limiting the area of impact if something goes wrong. Instead of launching to everyone at once, teams can roll out a feature incrementally, starting with internal users and then a small percentage of customers, and increasing exposure over time.

But feature flags and percentage rollouts alone aren’t experiments. They help teams manage risk but don’t stabilize audience assignments or provide statistically valid comparisons. They can catch a feature causing performance issues before it reaches your entire user base, but it won’t tell you why. And if conversion rates drop, there’s no guarantee the new feature was responsible—it could be a coincidence.

Without experimentation, teams risk making decisions based on incomplete data. They might roll back a feature that was an improvement or, worse, push something live that causes unseen problems down the line.

Experimentation should add value, not disrupt engineering workflows

Engineers often view experimentation as an add-on rather than a core part of the development process. They're typically focused on delivering code that functions without errors. However, that code is tied to a business goal: driving new customer behavior, helping customers solve problems, or driving other important business outcomes.

When it's time to bring a feature to market, shouldn't engineering be involved in the conversation about measuring its impact? In practice, engineers are often left out of the conversation, which has downstream impacts.

Separate tools = more headaches, slower decisions

One factor that worsens the situation is that the tooling engineers use to release features to market differs from the tooling used to measure that feature's impact.

Traditional experimentation platforms treat testing as an isolated process, disconnected from how engineers build and ship software. This disconnect introduces several challenges:

Context switching

Jumping between different tools interrupts engineering focus and slows development.

Manual configuration

Experiment enablement often requires extra steps outside standard engineering workflows. Differences between release and experiment management systems usually cause release challenges.

Rewriting code

Engineers often need to refactor or duplicate code to make a feature testable. Code written by experimentation tools does not follow company standards or understand and work with the entire codebase.

Delays in validation

Identifying data discrepancies between different systems can take significant time and only explains differences in how the tools work, not information about features released to the market.

As a result, many teams deprioritize experimentation due to the overhead, relying instead on instinct and anecdotal evidence, undermining the benefits of a data-driven approach.

STASH

Read: Full Stash case study →

Stash is used by over 4 million customers across the US. As a digital financial services company, Stash constantly seeks new ways to innovate and deliver unmatched customer experiences.

As part of this effort, Stash is building a culture of experimentation across the entire organization. For example, developers on the customer loyalty engineering team frequently run tests to measure the effectiveness of customer programs such as referral incentives, Stock-Back® Rewards, and more.

As the company continues to experiment across its platform, it has also instituted modernized software development practices. The software organization has pursued changes aimed at helping them release new functionality early and continuously, in keeping with Agile principles.

While Stash has made great strides in its experimentation and software delivery, LaunchDarkly helped Stash take things a step further by infusing experimentation across its entire stack.

Building experimentation into feature delivery

Rather than treating experimentation as a separate workflow, it should be a natural extension of how teams build, ship, and validate features. Teams can help eliminate friction and streamline testing by integrating experimentation directly into feature delivery.

How LaunchDarkly enables experimentation

LaunchDarkly removes workflow disruptions by embedding experimentation directly into the feature management process. This means engineers can:

- ✓ **Attach experiments to any feature**, measuring metrics tied directly to the flags controlling a release.
- ✓ **Run controlled tests without slowing down deployments**, helping to ensure experimentation enhances rather than hinders development velocity.
- ✓ **Ship and iterate in real time**, routing traffic to capitalize on the gains of a winning experiment immediately.
- ✓ **Create stable, statistically valid assignments** that can prevent bias and provide reliable insights.

By embedding experimentation within the development lifecycle, LaunchDarkly helps teams validate their hypotheses, mitigate risk, and optimize experiences, without sacrificing speed. Experimentation should be an enabler, not a blocker, for faster feature delivery.

"LaunchDarkly has played a big part in helping us build a culture at Stash, where we experiment with everything. It has also enabled us to release new features way faster than before. The fact that we can manage software releases and support experimentation in the same platform is remarkable."

Kahne Raja, Engineering Manager, Stash

STASH

Experiment types

Experimentation isn't one-size-fits-all. Different experiments answer different questions, and teams need the right approach for the right situation. In this ebook, we'll cover:

Bayesian and Frequentist approaches

Understanding different statistical methods and when to use them.

A/B testing

Direct comparisons between two variations to see which performs better.

Funnel optimization

Identifying drop-off points in a user journey and testing ways to improve conversion rates.

Warehouse-native experimentation

Measuring with trusted business data to enhance and deepen experiment results.

Each approach serves a different purpose, but they all share the same goal: helping teams make better product decisions based on real-world data.

2

A statistically rigorous toolkit—without the complexity

Experiments shouldn't require a statistics degree or a six-week setup cycle. Yet, many teams never get their experimentation programs off the ground because they feel too complex, too risky, or too disconnected from their daily work.

In this chapter, we'll explain what makes an experiment trustworthy and introduce the core principles LaunchDarkly builds into its Experimentation product so teams can move quickly and confidently.

Why do teams struggle to trust experiment results?

One of the biggest blockers to acting on experimental data is simple: doubt. What if the result is just random noise? What if our users weren't assigned consistently? What if we stopped the test too soon?

This hesitation is often justified but doesn't have to be permanent. LaunchDarkly can help teams reduce these sources of uncertainty by making statistical rigor the default, not a custom project.

What makes an experiment reliable?

A trustworthy experiment starts with the proper structure. It should be stable, in that users should be randomly and consistently assigned to a variation, with no mid-test switching. Conditions should be kept equal between groups, with only the tested variable changing. Teams should log every interaction they intend to measure. And lastly, experiments should have clear success metrics, with outcomes defined ahead of time.

Teams don't need to reinvent the wheel; LaunchDarkly bakes these fundamentals into the experimentation workflow.

What a reliable experiment looks like

Your product team wants to increase sign-ups by reducing friction on your onboarding page. You have a hypothesis: "Reducing the number of form fields will increase completion rates."

You spin up a flag to control two variations: one with the existing form and one with fewer required fields. Your metric? Completion rate for first-time visitors. Users are randomly assigned to a variation, and their experience stays consistent across visits. As data rolls in, you monitor performance using the LaunchDarkly results page.

A healthy experiment involves a straightforward question, a structured comparison, and built-in tools that help your team focus on learning, not logistics.

Understanding confidence without the confusion

If you've ever debated when to stop a test, you're not alone.

- "Can't we just release it now?"
- "How long do we need to keep testing?"
- "Do we need more users in the test?"

These are natural questions, and the answers depend on how confident you need to be. LaunchDarkly supports both **Bayesian** and **Frequentist** methodologies. Each offers different trade-offs around speed and certainty.

Use **Frequentist** methods when you need fixed-timeframe decision points and hard thresholds for rollout. Use **Bayesian** models when making decisions iteratively, watching probabilities update as new data comes in.

Common mistakes (and how to avoid them)

Stopping too early

A promising trend doesn't always mean a reliable result. Let the test run long enough for you to achieve confidence.

Inconsistent assignment

Your data won't reflect reality if users see different variations on different visits.

Unclear metrics

You can't learn from an experiment if you don't define what success looks like.

LaunchDarkly builds guardrails into every step of the process so these pitfalls don't become blockers.

3

Experimentation results: turning data into action

Many teams invest time and effort into setting up experiments, but they hit a wall when the results come in. Instead of delivering clear takeaways, experiments often lead to analysis paralysis—a situation where data is abundant, but the steps to follow are unclear. This can happen because:

- **Data is scattered**

  Experimentation results live in multiple tools, forcing teams to piece together insights from disparate sources.

- **Teams lack confidence in results**

  Without clear statistical interpretation, teams hesitate to act, fearing they might make the wrong call.

- **No structured decision-making process**

  When it's unclear who makes experiment decisions, findings are discussed but not implemented.

- **Experiments don't drive immediate action**

  Even when a winning variant is found, teams often struggle with the logistics of rolling it out quickly.

The results can include a lack of momentum for product teams, stalled progress, and wasted opportunities.

Not just data—actionable insights

To break free from analysis paralysis, teams need a direct path from results to action. That means:

- **Centralized results**

  A single, accessible source of truth that doesn't require engineers, product managers, and analysts to hunt for insights.

- **Clear, visual reporting**

  Data is presented in a way that immediately conveys meaning, not buried in raw spreadsheets or requiring complex analysis.

- **Confidence in the numbers**

  Statistical rigor should be built in, ensuring results can be trusted without needing deep statistical expertise.

- **Faster iteration**

  When a test concludes, teams should be able to immediately act on the findings without delays from manual deployment processes.

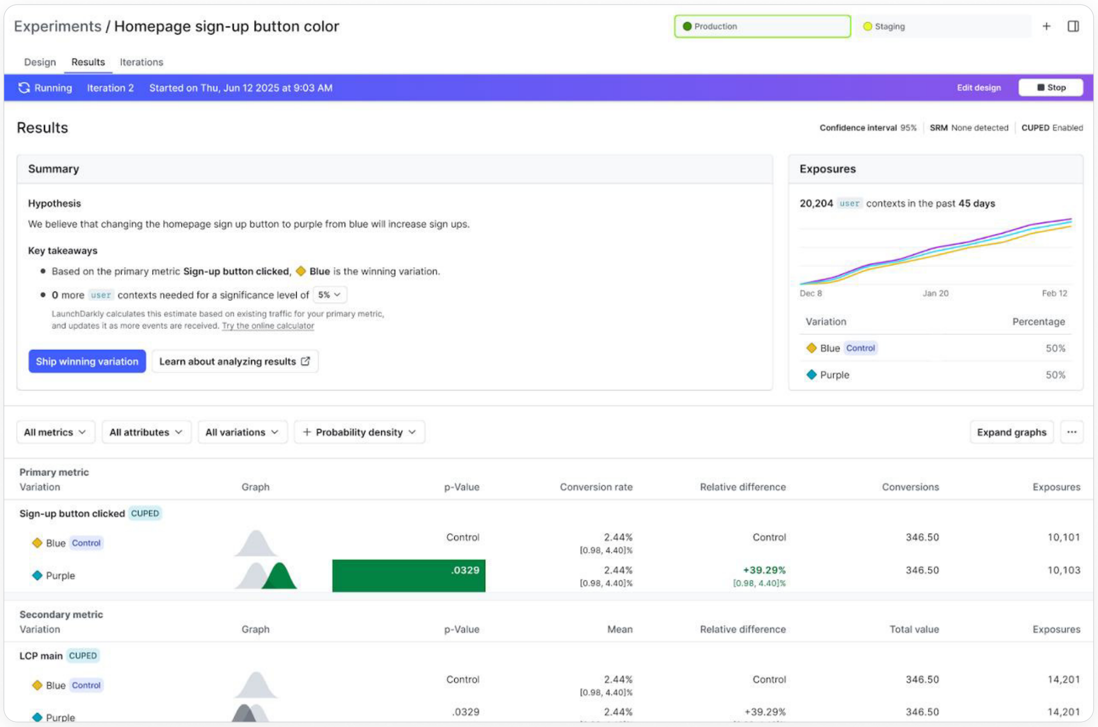
How LaunchDarkly delivers experimentation insights

1 Clear, centralized results; no more hunting for data

Many teams struggle with fragmented data, where experiment results are spread across analytics dashboards, BI tools, and engineering logs. LaunchDarkly solves this by keeping all experiment results in one place.

How it works:

- A unified results page displays exposures, conversions, and statistical confidence in an easy-to-understand format.
- Experiment history tracking allows teams to see how previous tests performed, preventing redundant experimentation.
- Configurable data sources help ensure you can efficiently source your data via streaming or warehouse. LaunchDarkly can meet your analysis needs where they are.



Results

Confidence interval 95% | SRM None detected | CUPED Enabled

Summary

Hypothesis

We believe that changing the homepage sign up button to purple from blue will increase sign ups.

Key takeaways

- Based on the primary metric **Sign-up button clicked**, ◆ **Blue** is the winning variation.
- **0** more `user` contexts needed for a significance level of 5%

  LaunchDarkly calculates this estimate based on existing traffic for your primary metric, and updates it as more events are received. Try the online calculator

Ship winning variation | Learn about analyzing results

Exposures

20,204 `user` contexts in the past **45 days**

Variation	Percentage
◆ Blue Control	50%
◆ Purple	50%

Primary metric Variation	Graph	p-Value	Conversion rate	Relative difference	Conversions	Exposures
Sign-up button clicked CUPED						
◆ Blue Control		Control	2.44% [0.98, 4.40]%	Control	346.50	10,101
◆ Purple		.0329	2.44% [0.98, 4.40]%	+39.29% [0.98, 4.40]%	346.50	10,103

Secondary metric Variation	Graph	p-Value	Mean	Relative difference	Total value	Exposures
LCP main CUPED						
◆ Blue Control		Control	2.44% [0.98, 4.40]%	Control	346.50	14,201
◆ Purple		.0329	2.44%	+39.29%	346.50	14,201

Example

A B2B SaaS company had been relying on a mix of spreadsheets, analytics dashboards, and BI reports to track the outcomes of their experiments. Every time a test concluded, a product manager would request data from analytics, pull historical metrics from engineering logs, and wait several days for a data scientist to validate the findings.

After moving to the LaunchDarkly experimentation platform, that workflow changed dramatically. With all exposures, conversions, and confidence levels available on one centralized results page, the team could review outcomes immediately without waiting for multiple teams to assemble the puzzle.

During a quarterly planning session, they quickly revisited results from three past tests using the experiment history view. They avoided re-running a test they had already tried (and that had failed) six months earlier. The time saved was equivalent to nearly two full weeks of work.

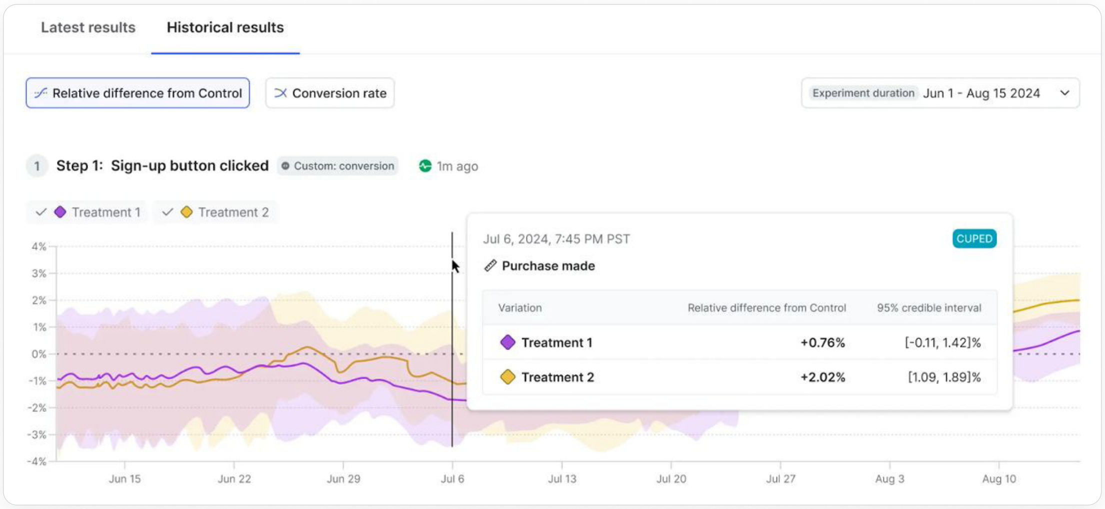
2 Easily analyzed results using visual analysis

Experiment results should be intuitive, not a wall of numbers that a data science team has to decode. LaunchDarkly presents clear visual insights so teams can quickly grasp what's working and what's not.

How it works:

- ✓ Conversion rate comparisons let teams instantly see which variation performed better.
- ✓ Attribute slicing gives teams the tools to analyze results across important customer segments (that you define) and to look for opportunities to capture margins.

✅ Time-series analysis allows teams to track experiment performance over time, detecting trends or anomalies.



Jul 6, 2024, 7:45 PM PST

Purchase made

Variation	Relative difference from Control	95% credible interval
Treatment 1	+0.76%	[-0.11, 1.42]%
Treatment 2	+2.02%	[1.09, 1.89]%

Example

A subscription meal delivery service ran experiments on a new homepage layout intended to boost sign-ups. Initial top-line metrics showed no significant difference between the two variations, leading the team to assume the new design was a neutral change.

However, when they used attribute slicing, they uncovered a different story: the new design actually improved conversion rates by 14% among mobile users, while desktop users saw a slight drop.

Using the LaunchDarkly time-series view, they also noticed that mobile conversions spiked during lunch hours—a pattern that hadn't shown up in aggregate metrics. This insight helped the team make a targeted decision: roll out the new layout to mobile only and refine the desktop version separately.

3 Statistical confidence that removes guesswork

Many teams hesitate to act on experiment results because they lack confidence in the data, not knowing whether the results are truly meaningful or just random fluctuations.

How it works:

- ✓ LaunchDarkly automatically applies Bayesian or Frequentist statistical models, eliminating the need for manual calculations.
- ✓ Results include clear probability statements (e.g., "Variant B is 89% likely to outperform variant A"), making decisions simple.
- ✓ Teams weigh predicted improvements against expected losses to make confident decisions.

Example

An online learning platform tested two versions of a course recommendation engine on its homepage. Variant B, the new algorithm, showed a modest lift in course clicks—around 3% higher than the control.

The team was ready to declare success. But the LaunchDarkly Bayesian analysis showed an expected loss of 2.4%, meaning that while Variant B looked better, there was still a meaningful risk that it could underperform in certain conditions.

Instead of a full rollout, the team ramped up gradually, limiting exposure to 25% of users while monitoring performance. This allowed them to validate the improvement without gambling the full user base. Two weeks later, with additional data, the probability of outperforming control rose to 96%, and the expected loss dropped below 0.5%. This gave the team the confidence to ship the complete course.

4 Results shipped in real time

Even when teams trust their experiment results, there's often a delay between knowing what works and deploying the winning variant.

How it works:

- ✓ Feature flags are integrated directly with experiment results, allowing teams to ship the winning variant without writing new code.
- ✓ If an experiment yields negative results, teams can immediately revert to the control group, without waiting for a full release cycle.

Example

A fintech startup was testing a redesigned loan application form aimed at reducing user drop-off. Midway through the experiment, Variant B showed a clear lead—users completed applications 22% faster with no drop in conversion.

Thanks to LaunchDarkly, the product manager didn't need to schedule a new deployment or wait for the next sprint. As soon as the team hit statistical significance, they used the integrated feature flag to ramp Variant B to 100% in real time, while the engineering team focused on other roadmap work.

Even better: a week later, when customer support flagged an increase in confusion among one niche user segment, the team simply reverted those users to control using audience targeting—without touching code or impacting the rest of the user base.

4

Attach an experiment to any feature, model, or AI configuration

Experimentation should be integrated into the development lifecycle, not used as a separate or additional step. Whether testing a UI change, backend model, or AI-driven recommendation system, embedding experimentation directly into rollout processes ensures that every release is an opportunity to measure impact and improve performance.

With LaunchDarkly, teams can:

- ✓ Attach experiments directly to the code's feature flags, ensuring that every change can be measured.
- ✓ Run controlled tests at any level, from UI components to backend logic to AI-driven personalization.
- ✓ Iterate faster by making real-time data-driven decisions, without redeploying code.

What is an A/B/n test?

A/B testing is the foundation of most digital experimentation strategies. It allows teams to compare two versions (or more in A/B/n testing) of a feature to determine which performs best. By randomly assigning users to different variations, teams can measure the impact of changes on key performance indicators such as:

- ✓ **Engagement**

  Does the new feature keep users interacting longer?

- ✓ **Conversion rates**

  Does the change lead to more sign-ups, purchases, or other key actions?

- ✓ **Revenue impact**

  Does this pricing experiment affect purchase behavior?

A/B tests provide a controlled and statistically valid way to evaluate whether a new feature should be fully deployed or iterated upon.

Creating a feature change experiment in LaunchDarkly →

What is a Funnel Optimization experiment?

A/B tests help you understand which variation performs better, but funnel optimization enables you to understand why users aren't completing the journey in the first place. These experiments are designed to uncover friction in multi-step workflows so teams can make precise, meaningful improvements.

- ? Where are users abandoning sign-up forms?
- ? Which step in the checkout flow causes the most drop-offs?
- ? How can we reduce time-to-conversion or increase completion rates?

Funnel experiments break the user journey into discrete stages, allowing teams to test targeted changes at the points where users get stuck.

Unlike traditional web-based testing tools, LaunchDarkly isn't limited to browser interactions. Your funnel steps can span the full stack—from mobile clients to server-side events—giving you a unified view of user behavior and conversion paths across platforms.

Running a funnel optimization experiment in LaunchDarkly →

Beyond UI changes: Experimentation for AI models and configurations

Modern experimentation goes far beyond front-end tweaks. With LaunchDarkly AI Configs, teams can safely ship, tune, and iterate on the AI models and settings that power their backend systems, creating safer deployments without introducing risk.

Here's how organizations are experimenting at the model and config level to unlock smarter user experiences and more reliable AI performance.

Recommendation models and ranking algorithms

Scenario: A retail platform wants to improve how products rank in search results and recommendation carousels.

Control

Uses a rule-based model sorted by popularity and category.

Variant

Tests a fine-tuned embedding-based model that uses user behavior vectors.

Success Metric

Increased add-to-cart rate and time spent browsing.

With AI Configs, the team can toggle between models or configurations and run controlled experiments in production to see which delivers better engagement.

Prompt and model parameter testing for AI assistants

Scenario: A SaaS company is scaling its AI-powered support chatbot and wants to fine-tune its tone and escalation behavior.

Control

A short, functional prompt with a low confidence threshold for escalation.

Variant

A longer prompt with an empathetic tone and a higher confidence threshold before escalating to human agents.

Success Metric

Higher customer satisfaction and fewer unnecessary escalations.

AI Configs let you iterate on prompts, temperature settings, and fallback logic in real time—testing changes without retraining models or redeploying backend code.

Real-time AI Config tuning for risk models

Scenario: A fintech company is updating its fraud detection model and wants to test new thresholds for flagging suspicious behavior.

Version A

Uses conservative thresholds to minimize false negatives.

Version B

Uses aggressive thresholds to reduce false positives.

Success Metric

Balance between fraud prevention and user experience (e.g., fewer blocked legitimate transactions).

Instead of hardcoding these thresholds, teams using LaunchDarkly can experiment with different configurations safely in production, using real-world data to validate their impact.

5

LEVERAGING YOUR DATA FOR DEEPER INSIGHTS

Warehouse-native experimentation and data export

Many organizations struggle to connect their experimentation results with core business metrics because data is spread across multiple tools. This disconnect creates problems such as:

Data Silos

Experiment results live separately from broader business analytics.

Inconsistent Metrics

Different tools define metrics differently, leading to confusion.

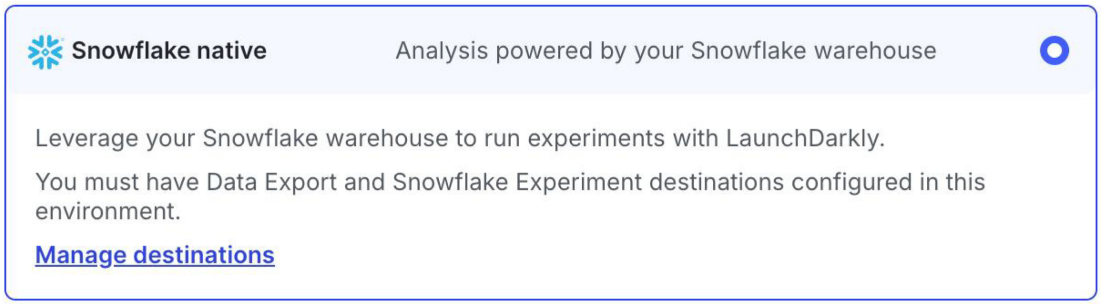
Delayed Decision-Making

Manually merging experiment data with business KPIs slows down actionable insights.

LaunchDarkly solves these challenges by enabling Warehouse-Native Experimentation with Snowflake and offering data export experiments for deeper analysis in external tools.

Warehouse-Native Experimentation with LaunchDarkly and Snowflake

Warehouse-Native Experimentation integrates LaunchDarkly's experimentation data directly into Snowflake's AI Data Cloud, allowing teams to analyze experiment results alongside business data in Snowflake. By keeping all analysis within the data warehouse, teams can use trusted organizational metrics to measure experiment outcomes and reduce data movement.



Snowflake native — Analysis powered by your Snowflake warehouse

Leverage your Snowflake warehouse to run experiments with LaunchDarkly.

You must have Data Export and Snowflake Experiment destinations configured in this environment.

Manage destinations

Example

A subscription-based streaming service wanted to evaluate whether a $2 price increase on new subscriptions would increase revenue without driving churn. But like many data-savvy organizations, they already had well-defined metrics —such as monthly revenue per user and 60-day retention—living in their Snowflake warehouse.

Rather than re-instrument their app to send new event data into LaunchDarkly, the team used Warehouse-Native Experimentation to connect LaunchDarkly directly to those existing metrics. No new pipelines or duplicated tracking; just analysis run against data that the business already trusted.

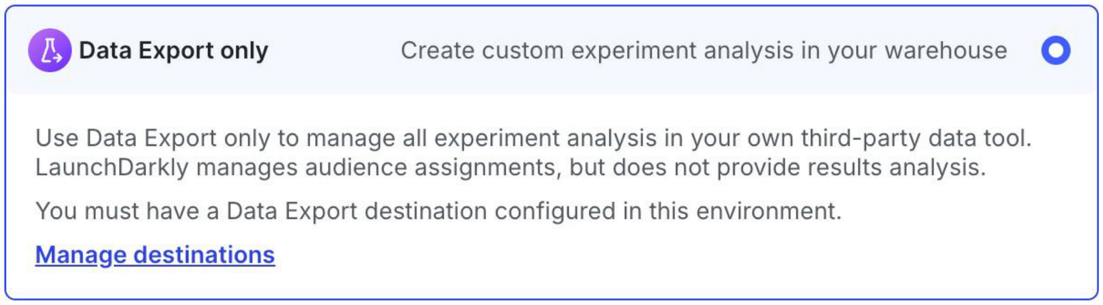
As the experiment ran, LaunchDarkly measured revenue impact and retention automatically using those warehouse-defined KPIs. Within a few weeks, results showed a 9% boost in revenue, but also a notable drop in retention for certain user segments. Confident in the results—because they were based on their own source-of-truth data—the team moved quickly. They rolled back the price change for most new users while keeping it in place for a high-LTV segment that showed no negative retention signal.

The team avoided unnecessary engineering work, made a faster decision, and unlocked a more nuanced pricing strategy by using metrics they'd already invested in and data that never had to leave their warehouse.

Data export experiments: using external tools for custom analysis

Unlike standard experiments that require defining metrics in LaunchDarkly, Data Export Experiments send raw experiment data to external analytics tools. This enables teams to:

- ✓ Analyze results in BI tools and data warehouses such as Snowflake or BigQuery
- ✓ Use custom statistical methods beyond LaunchDarkly built-in Frequentist and Bayesian models.
- ✓ Maintain metric consistency by using existing business definitions.


Data Export only Create custom experiment analysis in your warehouse ●

Use Data Export only to manage all experiment analysis in your own third-party data tool. LaunchDarkly manages audience assignments, but does not provide results analysis.

You must have a Data Export destination configured in this environment.

Manage destinations

Example

A B2B SaaS company was rolling out a new in-app automation feature designed to reduce manual workflows for enterprise admins. The product team believed this feature would increase long-term customer satisfaction and reduce churn, but the impact wouldn't show up in simple conversion metrics alone.

Instead of relying on built-in analysis for short-term engagement, the team set up the experiment in LaunchDarkly as a data export experiment, streaming variation assignments and exposure events directly into Snowflake.

From there, the data science team combined experiment data with downstream signals—support ticket volume, usage frequency, renewal status—and fed it into an existing machine learning churn risk model.

The experiment highlighted a nuanced insight: while the feature had little impact on daily usage, it significantly reduced churn risk for admins in accounts with over 50 seats. This information wouldn't have surfaced in top-level metrics, but it was critical for retention strategy.

Armed with this evidence, the company fast-tracked full rollout to high-seat customers and started a follow-up experiment to explore the impact on mid-sized accounts.

Benefits of Warehouse-Native and data-export experimentation

Warehouse-Native and data export experimentation with LaunchDarkly can provide:

- ✓ **A single source of truth**

  Use existing business data instead of maintaining separate experiment tracking.

- ✓ **Greater analytical flexibility**

  Apply custom statistical models using BI tools.

- ✓ **Faster decision-making**

  Experiment results appear directly in Snowflake, BigQuery, or other platforms for immediate insights.

- ✓ **Scalability**

  Run large-scale experiments without performance impact on application infrastructure.

6

From instinct to insight-driven development

Experimentation isn't just a process—it's a mindset shift. By embedding experimentation directly into feature delivery, teams can:

Innovate faster

Reduce guesswork and ship impactful features quickly.

Minimize risk

Test changes before full rollout, ensuring a smooth user experience.

Optimize outcomes

Continuously refine product strategies based on real-world data.

Break down silos

Enable collaboration between engineering, product, and data teams.

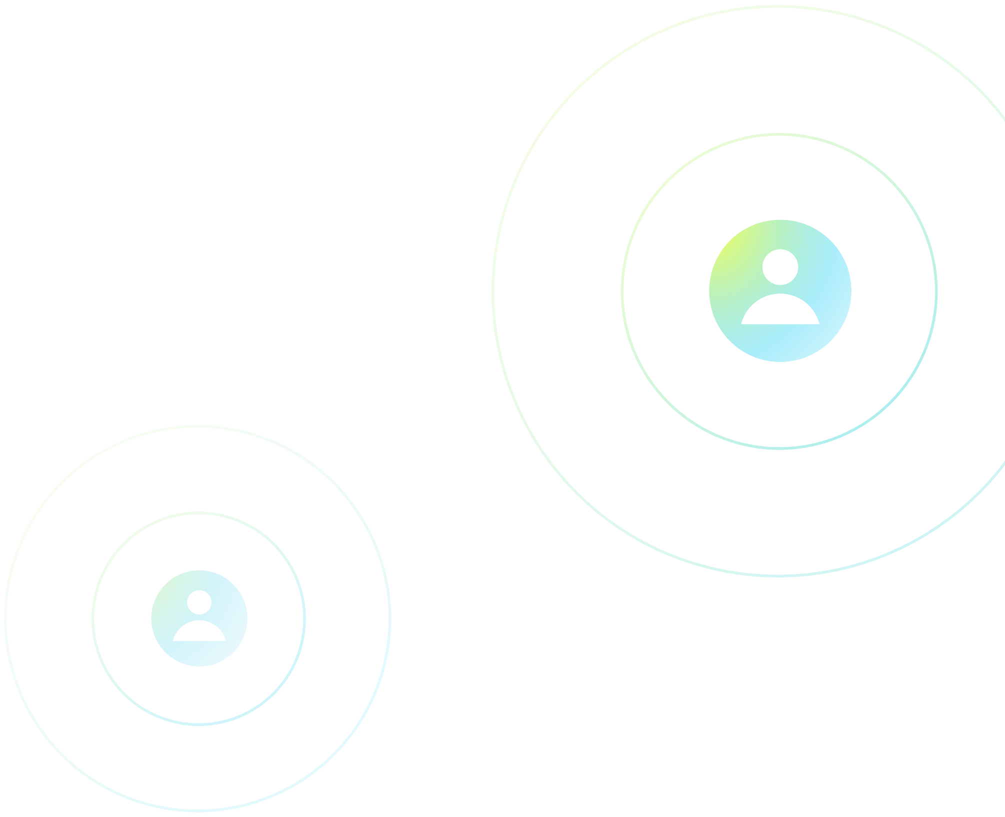
Why use LaunchDarkly Experimentation?

Many teams struggle with experimentation because traditional platforms operate in isolation from the software delivery process. LaunchDarkly embeds experimentation directly into feature flags for easier testing. It supports everything from simple A/B tests to AI model evaluations and allows teams to use both Bayesian and Frequentist approaches. It also ensures stable user assignments, statistical rigor, and real time insights.

Teams can learn iteratively every time they ship a feature. Every release is an opportunity to:

- ✓ Validate assumptions and iterate based on evidence.
- ✓ Discover new user behaviors that inform future development.
- ✓ Create improvements across all aspects of the business.

The future of software development belongs to teams that learn faster. Are you ready to start experimenting?



Next steps

Let’s turn every release into a learning opportunity. The next breakthrough starts now.

Get in touch

Get a demo

launchdarkly.com | sales@launchdarkly.com