

LaunchDarkly →

3 ways to end the misery of tech migrations

Safely migrate and modernize across
cloud services, databases, and APIs.

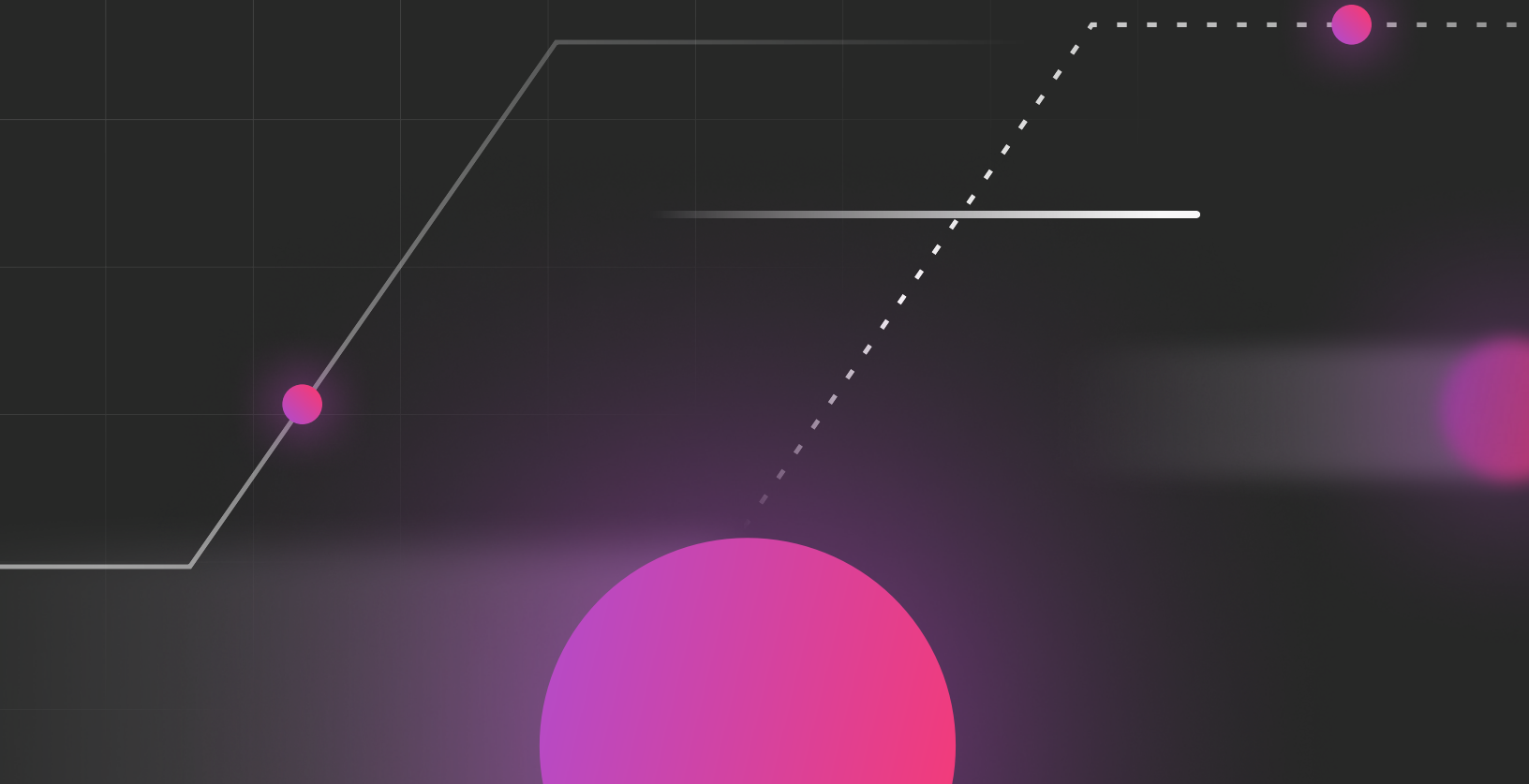


Table of Contents

Introduction	2
01 3 major causes of migration failures	3
Big-bang migrations lead to big-bad results	
You can't fix what you can't see	
Back-end errors tank customer experiences and drive up costs	
02 How LaunchDarkly can help	6
Avoid nightmare "big-bang" migrations	
Validate performance across each step of a migration	
Recover faster from failure	
03 Customer success stories	19
Nestlé Purina's Petfinder	
Hireology	
04 Get started	24

The pain of modernization

Adopting new cloud services, databases, and APIs is a constant in modern software delivery, but technology migration projects are expensive, time-consuming, and fraught with risk. A [2021 report from Gartner](#) found that 32 percent of cloud initiatives fail. Why is modernization so painful?

- ❗ It's difficult to break migrations down into smaller chunks, which forces businesses to undertake risky, big-bang migrations.
- ❗ The prospect of exposing new back-end services to all users is scary due to the challenge of testing and monitoring the impact at a realistic volume and scale.
- ❗ Back-end errors can quickly tank customer experiences and SLAs and drive up costs.

These challenges deter organizations from migrating and modernizing legacy systems. But the risk of inaction is equally great, potentially resulting in the type of [outages experienced by Southwest Airlines](#) in the past year.¹ Organizations need a safer, faster, more predictable way to evolve their tech stack to stay ahead of competition.

LaunchDarkly lets organizations modernize technologies faster by introducing new application components in a gradual and controlled manner. In this guide, we'll explore how you can break migrations down into manageable pieces, monitor performance and business metrics to ensure migrations go as planned, and avoid negative customer impact by resolving errors immediately.



3 major causes of migration failures

Before we can succeed with migrations, we need to understand why they fail.

01

Big-bang migrations lead to big-bad results

In 2018, a UK-based bank went through a botched database migration, losing an estimated £330 million and 80,000 customers in the process². The story serves as a cautionary tale for businesses attempting to perform migrations at a scale that quickly becomes unmanageable when things go wrong.

While we don't know what exactly happened inside the bank, we *can* say that many organizations are forced to migrate large swaths of infrastructure all at once. For example, a database migration doesn't just entail switching databases; it may also require changing API routing, front-end services, and other parts of the infrastructure. These big-bang migrations cause a great deal of stress and present major risks.

02

You can't fix what you can't see

Rolling out new back-end components to all users all at once is quite risky. If organizations could release to a subset of users and environments first, thus limiting the blast radius of potential errors, it would mitigate some of that risk (spoiler: feature flags shrink the blast radius). But this alone isn't enough to ensure a smooth, successful migration.

Organizations also need a way to monitor the impact of infrastructure changes on application performance and business metrics at every stage of a migration. Unfortunately, many organizations lack a way to gradually roll out new back-end systems. And they struggle to easily correlate a degradation in performance with a specific back-end change.



03

Back-end errors tank customer experiences and drive up costs

When errors occur during a migration, they often take a long time to resolve. In many cases, teams have to roll back to an older version of their service. This type of recovery can be quite burdensome and time-consuming. Worse, it fails to shield customers from the impact to services. In the UK bank example above, millions of customers were unable to access their accounts and, in some cases, logged in to find other customers' private account information instead of their own. Failures like these can cost businesses gobs of money.

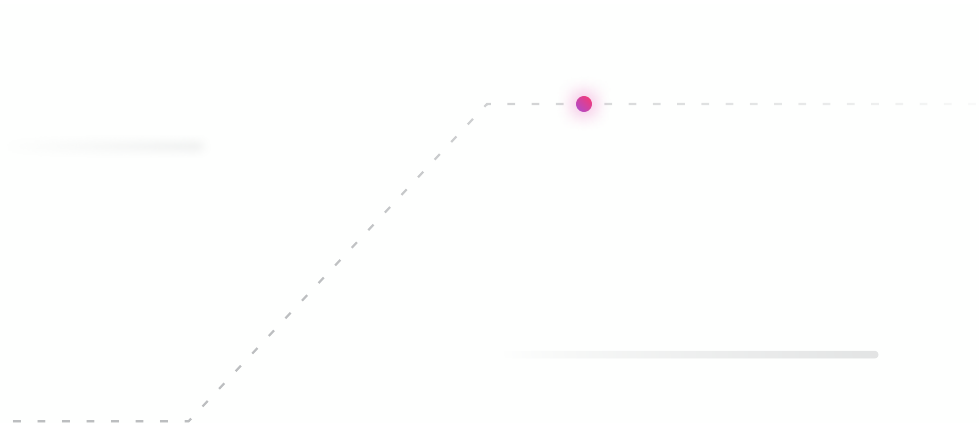
The above challenges make organizations apprehensive about pursuing modernization. But failing to adapt is equally risky, and companies that do embrace modernization stand to gain.

Thankfully, it *is* possible to mitigate migration risks and reap the rewards—let's explore how.

- ¹ The December 2022 outage led to [an estimated loss of over \\$1B in revenue](#).
- ² [TSB attempted to migrate 1.3 billion customer records](#) from a system run by its former parent company to a new database.



How LaunchDarkly can help



With LaunchDarkly, you can confidently migrate and modernize your tech stack without enduring repeated failures and chronic migraines. In this section, we will explore three chief benefits of LaunchDarkly that enable you to overcome the modernization challenges outlined above.

Jump straight to a benefit section:

#1 [Avoid nightmare “big-bang” migrations](#)

#2 [Validate performance across each step of a migration](#)

#3 [Recover faster from failure](#)

01

Avoid nightmare “big-bang” migrations

Migration projects are daunting and full of risk, and when tackled all at once, they often require maintenance windows in which businesses can't operate. LaunchDarkly enables teams to mitigate those risks by introducing new infrastructure, databases, APIs, and other services in a progressive, controlled way to avoid outages and maintenance windows. By approaching technology migrations in digestible chunks, you reduce the likelihood of disaster.



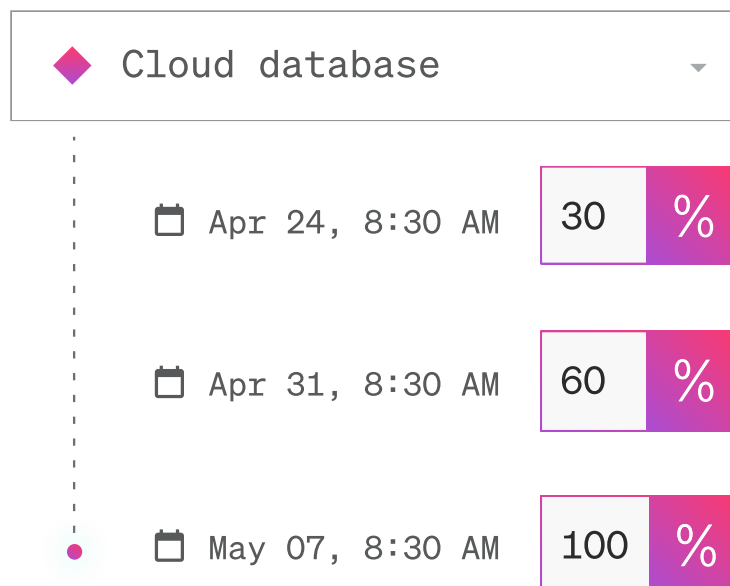
Break migrations into manageable pieces

Migrating to a cloud provider like AWS sounds like one big project, but it actually involves moving a lot of pieces, such as infrastructure, databases, and services. Flicking the switch on all of those at once is both intimidating and risky. LaunchDarkly enables you to isolate the different pieces, so you can introduce new infrastructure, databases, APIs, and other services gradually to avoid outages and maintenance windows.

With LaunchDarkly feature flags, you can treat a specific back-end service like a feature. Let's imagine you're migrating infrastructure, and it involves multiple back-end services: databases, APIs, etc. You can wrap a new database in a feature flag and gradually route traffic away from the legacy database to the new system—in the same way you would with a feature. And you can do the same with APIs and other back-end services.

Atomizing the migration in this way reduces risk.

Progressive rollout



Feature flags

Feature flags are a software development process used to enable or disable functionality remotely without deploying code. With feature flags, you can deploy changes to a production environment without making them visible to users (i.e., decouple *deploy* from *release*). This reduces risk when shipping software changes, whether those changes are front-end features or back-end components.

When code deployments and releases (i.e., code changes that are accessible to end-users) are decoupled, enterprises have more control over the migration of large and highly complex systems. Teams can push code into production to release new systems, while gradually shifting users off the legacy platform.



Watch: [Use Feature Flags to Avoid Downtime During Migrations](#)



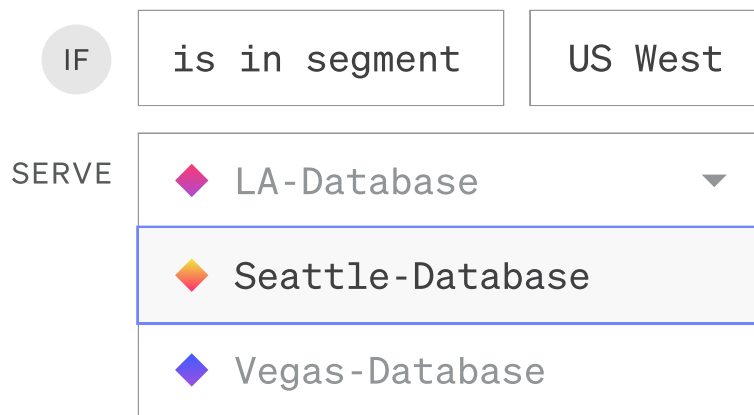
Go beyond infrastructure routing

Control migrations at the application layer to flexibly target any parameters. Let's look at a concrete example. Imagine a scenario in which developers want to incrementally roll out a new cloud database system by region: starting with the U.S., then moving to Europe, then to Asia, and so on. If they rely exclusively on infrastructure tools such as a load balancer to perform that action, it could needlessly prolong the process. Developers may end up submitting a ticket to the IT operations team every time they need to add a destination to the load balancer or increase the percentage of traffic to the new database. And developers lose control over the migration in the process.

A better alternative is to use flags alongside your infrastructure tools. In this example, a developer could submit a single ticket to enable an open



connection between their API/front-end and the new database system. Then, the developer(s) could alter the traffic-routing rules themselves with feature flags. And, of course, as with any feature-flagged change, if during one stage of the migration, latency increases or error rates surge, they could instantly revert back to the previous stage—one in which operational health was stable.



Segment experiences

Use sophisticated targeting rules to expose certain back-end services to specific user segments based on business needs. With LaunchDarkly's [custom contexts](#), you can experiment and conduct staggered migrations based on parameters that map to your business needs.

You can enable a new API for just internal developers for testing, while disabling it for everyone else. Or you can route certain types of web requests to a new cloud database while routing other requests to your legacy system. Or you can enable your new cloud infrastructure for users on Windows operating systems while supporting other operating systems with your on-premises data center.



LaunchDarkly targeting gives you virtually unlimited flexibility, ensuring your migration meets your business requirements perfectly.

Custom contexts

Custom contexts are a capability in LaunchDarkly that enables you to define, experiment, and target based on your data model, business language, or release process. So, instead of only being able to target users, with custom contexts you have almost endless flexibility to deliver to any kind of thing and as many kinds of things as you like, all at once. For example, you can enable a new back-end service for only users in Philadelphia, on iPads, whose mother's maiden name is Stallone. Everyone else gets your legacy infrastructure.



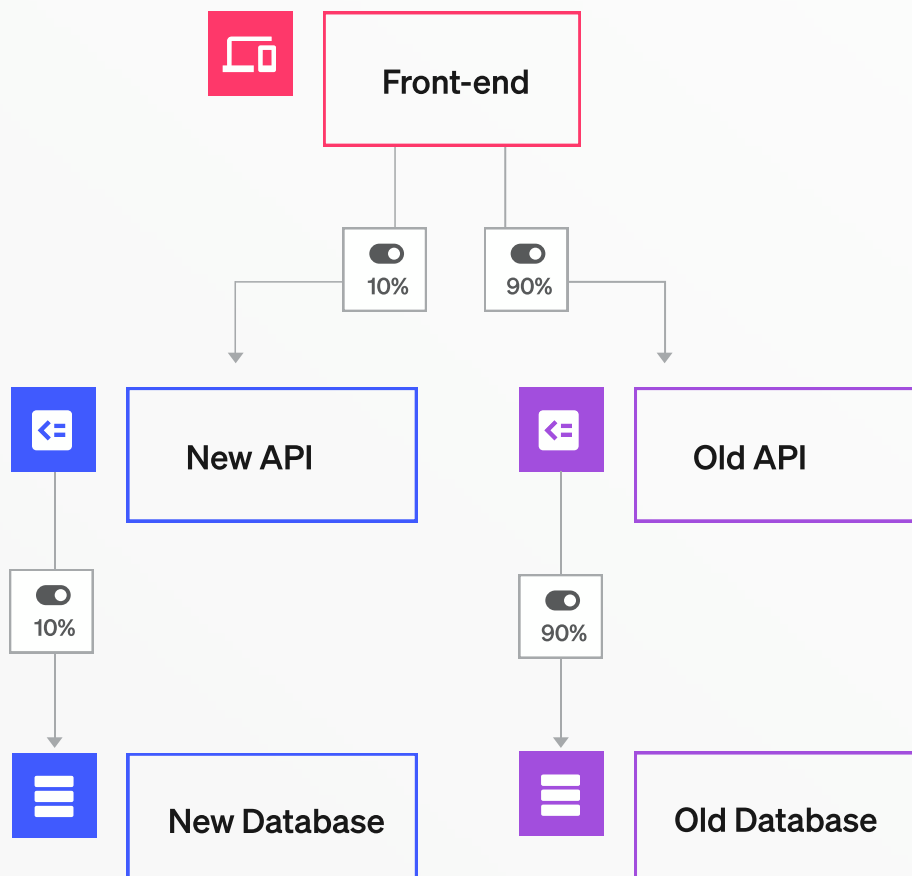
Watch: [Migrating to Custom Contexts](#)



EXAMPLE

Progressive rollouts give you control and flexibility while migrating

You have a three-tier application composed of a front-end, API, and database. You're migrating to a new API to address some performance concerns with the existing one, so for some time, your front-end may be communicating with two APIs at once. If you rely on traffic rules to enable testing of the new API, your users are going to be impacted if something goes wrong. Instead, you could use a feature flag and targeting rules to route some percentage of traffic to the new API—if something goes wrong, you can immediately flip the flag to direct traffic to the working API. If all goes well, you might roll out a new, cloud-based database, progressively migrating workloads for the new API to your new cloud database.



You can use feature flags to progressively roll out new APIs and/or databases in parallel with legacy ones, and fail-over quickly in the event of issues with your new services.

02

Validate performance across each step of a migration.

Instead of having to roll out new back-end components all at once and wait for frustrated customers to alert you to an error, LaunchDarkly provides full visibility and governance around critical changes to keep your teams aligned and your business, and data, safe.

Progressively release new services

So, you've broken your migration down into less risky parts. Now, you can [introduce new back-end components to users incrementally](#) to limit the blast radius of potential errors. You can also put guardrails in place to prevent unintended consequences by leveraging LaunchDarkly [Approvals](#). This gives more team members the chance to provide input on planned changes to a flag.

Here's what it might look like, for example, to use feature flags to gradually roll out a new database. At the start of the migration, you can route 100% of 'read' and 'write' events to the old database, while duplicating 10% of the write events and then pushing them to the new database. As you monitor performance and see that your system has remained stable, you can then move to the next phase of the rollout, until 100% of events are funneled to the new system.

Similarly, you could progressively deliver a new API or some other service by device type, geolocation, or any other attribute you can imagine. You could start with web browsers, then gradually expand to iOS devices, then to Android devices, then to gaming consoles, and so on. And you could do so at your own pace.



These are great first steps, but once your migration is under way, how do you monitor and measure how the migration is progressing? How do you know that it's having the intended impact?

Connect innovation to business value

If you're migrating to a new service or database, you should be able to measure how those changes impact app performance and business metrics, and adjust accordingly.

With LaunchDarkly, you can create back-end experiments to collect data on which version of your application has the desired impact. Whether you're looking to improve page load times, reduce server costs, or increase revenue, you can measure the effect of your change and immediately ship the best variation to your audience.

As an example, operations and platform engineering teams can use LaunchDarkly to create a back-end experiment that measures the impact of a new API on infrastructure costs (from AWS, Microsoft Azure, etc.) They can measure the costs against a baseline and tweak the API implementation until the costs fall to a reasonable level. In this case, they will have migrated to a new API while driving operation costs down, thus benefiting the business on two fronts.



Read: [How to Enable Server Side Experimentation](#)



Monitor via APM tools

Sometimes, even the most sensible and controlled migrations don't go as planned. The key in these instances is to have visibility: first you need to know that something isn't working, you need to isolate the cause of the error, and then immediately mitigate the impact. By integrating LaunchDarkly with an application performance monitoring (APM) tool, you can quickly uncover changes or issues caused by a migration.³



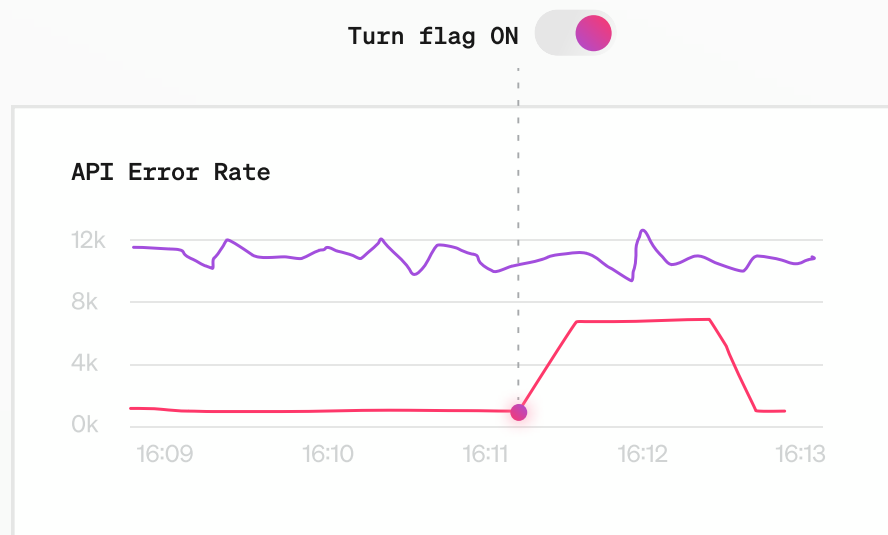
Watch: [LaunchDarkly and Datadog](#)



EXAMPLE

Datadog + LaunchDarkly make troubleshooting easy

Let's say one of your developers just initiated part of a migration and is controlling it with a feature flag. Suddenly, error rates are surging, but you're ready for it because with the LaunchDarkly and Datadog Events integration, a) your ops team can see the spike on their Datadog dashboard and b) the team can quickly correlate the error with the developer's feature flag, which had recently been activated. Now, resolution is as simple as shutting off the flag tied to the back-end change that caused the error, so the team can debug without impacting customers.





New Database



03

Recover faster from failure

Your customers never have to feel the impact of migration errors. With LaunchDarkly's enhanced recovery capabilities like automatic flag triggers and kill switches, you can recover from any errors in near-real time without your customers ever knowing.

Revert to safety instantly

In the event of an error or performance issue during a migration, teams can perform a controlled fail-over back to the original service or infrastructure within milliseconds by [using a feature flag as a kill switch](#). As in the example above, if part of a migration causes an issue, simply toggling the flag off again is enough to restore service to normal, without having to roll back or deploy new code.

Feature flags allow you to deactivate a broken or problematic service in runtime. For example, imagine you've successfully routed 25% of all web traffic to a new API. But when you expand to 50%, it causes your website to malfunction. If you're using a flag to govern the rollout, you could disable the flag and instantly return to the stage in which only 25% of traffic hit the new API. Or out of an abundance of caution, you could turn the API off entirely. The key point here is that you can resolve this error without manually re-routing traffic by updating the destination or weights on a load balancer. Instead, you revert the flag to a stable API with the flip of a switch.

Empower more teams to resolve incidents

Another advantage of not having to deploy code to remediate migration issues is that you can enable more team members to resolve incidents in real-time. LaunchDarkly's custom roles allow you to define permissions for feature flags, projects, environments, metrics, and teams.

With the right access controls in place, dev teams can toggle broken functionality off without waiting for others to respond or having to run a change request through a cumbersome approval process. Additionally, your SRE or support engineering team is empowered to remediate incidents without having to wait for the relevant developer to be paged to address the root cause.

//

Rather than calling someone at three o'clock in the morning, we'll just shut the flag off. We'll figure it out in the morning, please stay asleep.

Justin Duhatschek

Progressive Delivery Manager at KBX Technology Solutions

³ LaunchDarkly integrates with APM tools such as Datadog, Honeycomb, New Relic One, AppDynamics, Dynatrace, and more. See all our [APM and observability integrations](#).



Automate remediation

Beyond human intervention, the next level of risk mitigation for migrations is automation. LaunchDarkly's integrations with popular APM providers mean that you can set error rate thresholds and leverage feature flags as migration circuit breakers ([flag triggers](#)) to instantly recover when the threshold is exceeded.

EXAMPLE

Automate error remediation with APM flag triggers

As before, when migrating a back-end component of your app, you can wrap it in a feature flag to track its performance and behavior, and use a flag trigger to connect the feature flag to a performance metric.

A flag trigger is composed of an unguessable URL that LaunchDarkly assigns to you. Accessing one of these URLs either turns the flag on or off. You can connect these URLs to an alert in your APM, observability, logging, or error tracking tool, or any other tool which can fire webhooks.

When a flag trigger is connected to one of your tools and the performance of, say, your new algorithm crosses a predefined threshold, your tool triggers an alert that hits the URL and turns the feature flag off (so no one has to get paged to hit the button).

Create trigger



Datadog events



Dynatrace



Honeycomb



New Relic One



Customer success stories

The best migration stories are free of drama. Here are two examples of customers leveraging LaunchDarkly's safe, progressive migration capabilities to modernize their legacy applications without downtime for their customers.



Nestlé Purina's Petfinder migrates to microservices without disrupting service for pet seekers

Challenge

[Petfinder](#) is the largest online pet adoption site in North America. The site was acquired in 2013 by Nestlé Purina Petcare, a subsidiary of Nestlé S.A, a multinational food and drink processing corporation. Since then, Petfinder and Nestlé Purina Petcare have been on a mission to match people with their perfect pets, but Petfinder's legacy on-premises monolithic website hindered agility and created deployment risk. They needed to re-architect to microservices while maintaining business continuity.

Solution

With LaunchDarkly, Petfinder could gradually migrate their infrastructure piece by piece, while keeping their site up for pet seekers. LaunchDarkly enabled Petfinder to roll new front- and back-end features out to a subset of users, validate performance along the way, and instantly roll back if needed.



Watch: [Nestle Purina and LaunchDarkly](#)



Results

- ✓ Migrated to modern architecture without downtime
- ✓ 2x increase in deployments
- ✓ Improved DORA metrics
- ✓ More efficiency and confidence around releasing front-end and back-end capabilities

//

As we're continuing to modernize all of the technology that we're using, we're able to use LaunchDarkly on AWS to roll out new features in a safe way, to make sure that we don't take the site down ... LaunchDarkly gives us the confidence to make these changes that make it more efficient, because we want to make sure that we can have the best experience for all of our users.

Darla Ahlert

Solutions Architect, Petfinder (Nestlé Purina Petcare)





Hireology migrates seamlessly to Azure Cloud and slashes infrastructure hosting costs

Challenge

[Hireology](#) is an all-in-one hiring and HR platform founded in 2010. Hireology's development team was struggling to deliver successful, frequent releases, while also facing the challenge of a large-scale migration to Azure.

Solution

Hireology broke down their monolithic migration into components to reduce risk, migrating each individually to Azure. They then leveraged segmented user targeting to route user traffic between their legacy systems and public cloud. With this approach, they were able to revert instantly in the event of an error.

Results

- ✓ 8% change failure rate, down from 75%
- ✓ 14x increase in deployments and 65% increase in loading time speed
- ✓ Migrated legacy components to Azure Cloud seamlessly and with no negative impacts on customers
- ✓ Significantly reduced infrastructure hosting costs with LaunchDarkly



Additionally, by integrating LaunchDarkly with APM (Datadog), Hireology:

- ✓ Increased deployment frequency from 2x a month to on-demand
- ✓ Cut lead time from 14 days to 3 hours
- ✓ Slashed recovery time from a couple days to less than an hour

//

What LaunchDarkly does is it provides a safety net for this change so we can test out the Azure app services without exposing it to everybody in the world.

Nick Macellaio

DevOps Manager, Hireology



Read: [How Hireology Enables Faster, Safer Software Delivery and Migration](#)



Get started

With LaunchDarkly, you can leverage the benefits of modern technologies faster by introducing new application components in a gradual and controlled manner. By monitoring and validating performance across each step of the modernization journey, teams can resolve errors instantaneously and ensure quality customer experiences.

If you'd like a more in-depth understanding of how to implement LaunchDarkly for the purpose of migrating and modernizing your tech stack, then request a customized, personally-guided demo.

Ready to get started?

[Get a demo](#)

