

LaunchDarkly ➔

LaunchDarkly Pro Tips & Tricks: Volume One

Toggling, Flag Triggers, Flag Prerequisites, and the Relay Proxy

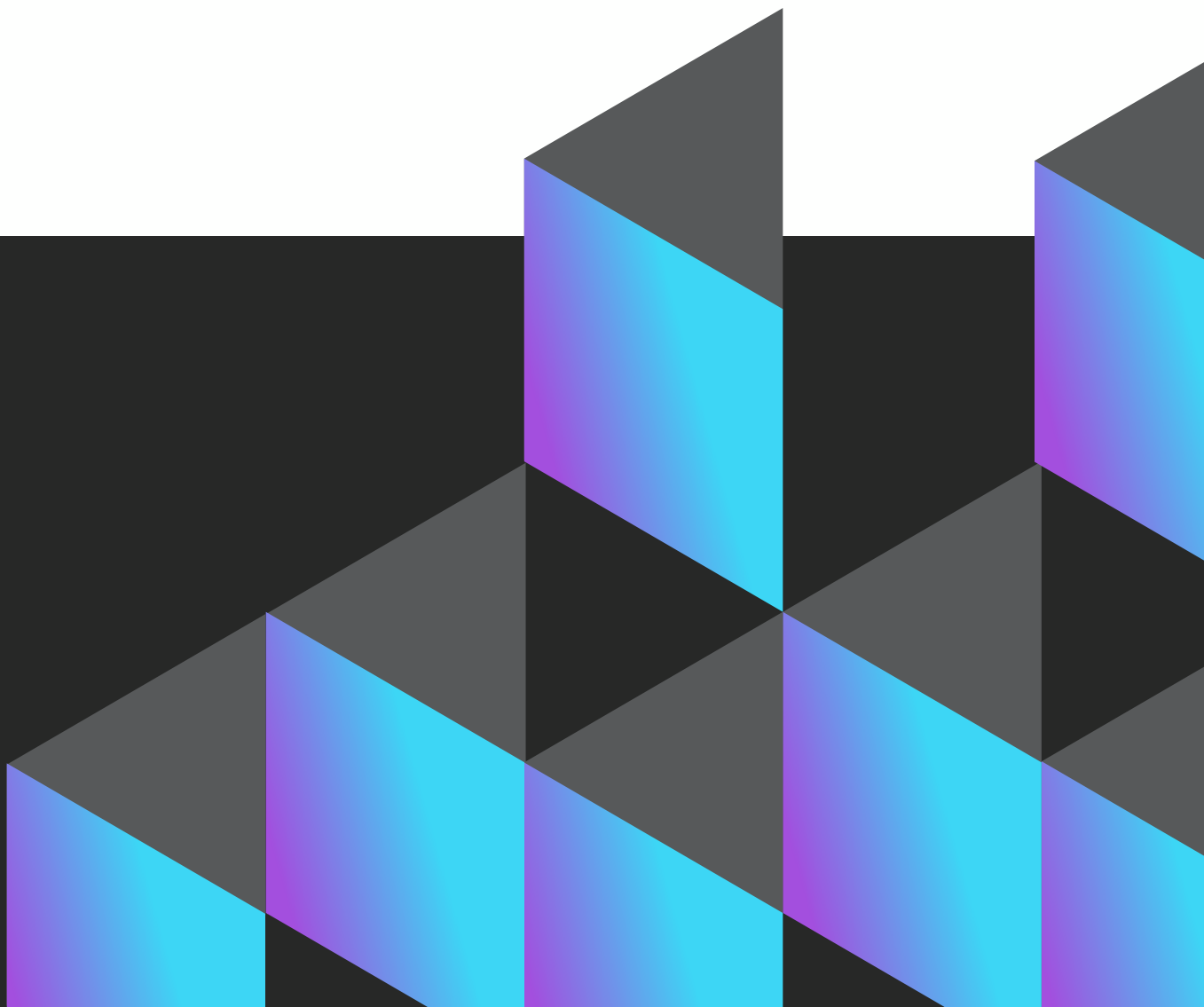


Table of Contents

Introduction	01
Advancing Your Toggling	02
The Workflow Builder	03
Flagging Triggers	10
Relay Proxy at LaunchDarkly	14
Deploying the Relay Proxy with Spinnaker	16
Automating upgrades with the Relay Proxy	16



Introduction

So, you've gotten used to working with LaunchDarkly and now you're looking for some new ways to leverage the full scope of the features and capabilities at your disposal.

Great! Welcome to the first volume of our series on LaunchDarkly tips and tricks to help you do just that.

In this edition, we'll be covering:

- Advancing Your Toggling
- Flag Triggers
- Flag Prerequisites
- How to Use the Relay Proxy

We'll kick things off with a proper overview of the [Workflow Builder](#) and go from there.



01 Advancing Your Toggling

In this section, we're going to walk through a few of the most common ways we see people extend their toggling beyond just hitting the switch in the UI. This will give you some visibility into the different ways you can enable or disable features within your environments.

Let's get into it.



The Workflow Builder

The [Workflow Builder](#) is a favorite at LaunchDarkly, mainly because we're huge fans of features that allow developers and operators to recapture some of their time. Put simply, the Workflow Builder allows teams to scaffold out process-driven workflows for releasing feature flags in LaunchDarkly.

There are a few key capabilities of the Workflow Builder that are worth highlighting.

Progressive Rollouts

Progressive rollouts are an out-of-the-box capability. With the click of a button, you can automatically plot out a progressive rollout that gradually increases by a configurable percentage over a defined period of time. For example, this enables you to roll out a change to 10% of your users each day over the course of 10 days. If there's a problem along the way, the process can be canceled and the feature can be disabled immediately.

Configure progressive rollout

Tell us the roll out increments and frequency, and we'll do the rest.
[Learn more about progressive rollouts.](#)

Start workflow on
2022-03-28, 21:21 -07:00

Roll out variation
true

Increments ⓘ
10%

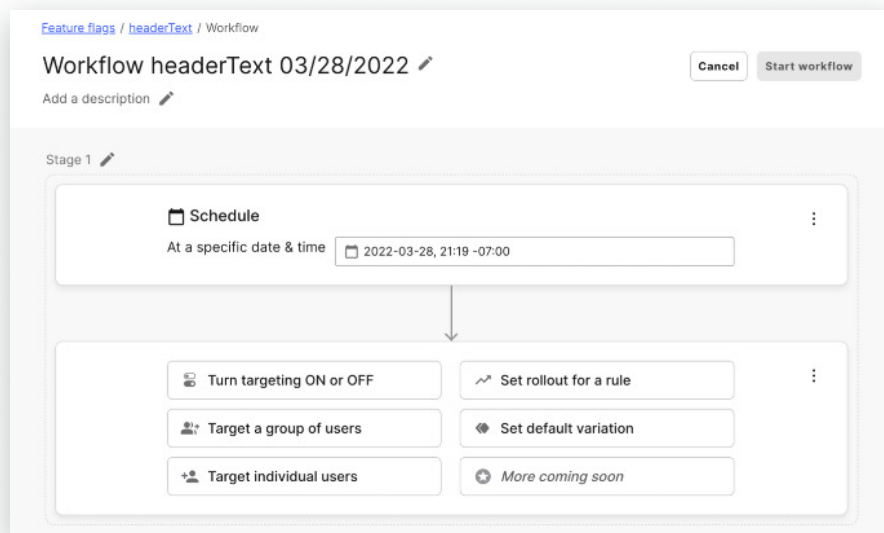
Frequency ⓘ
day

BackReview workflow



If you want a more customized approach, the Workflow Builder allows you to plot out a custom workflow with the specific steps you'd like to execute. For instance, you can create a workflow that requires approvals from specific people within your organization or a specific date to begin executing.

From there, you can have the Workflow Builder create multiple steps around enabling or disabling a feature, adding targets to a feature flag, changing the default variations for a rule, or even building out specific rollout percentages (more on that below).



When coupled with scheduling out flag changes, these rules give you the ability to plan out feature releases that impact specific teams.

- Other example use cases for flag targeting:
- Targeting a mobile-optimized UI for testing by IOS mobile devices
- Ensuring that Internet Explorer browsers receive a more compatible user interface

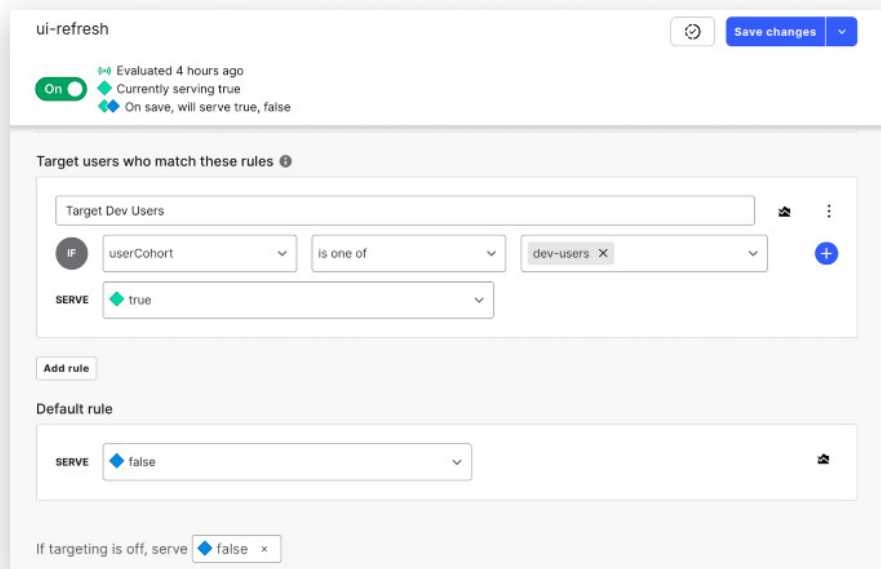
Allowing developers who have a specific header configuration on their API requests to receive a new version of API for early testing, or bypass a maintenance mode on a development API



Targeting rule-based rollouts and scheduled changes

One of the biggest superpowers of LaunchDarkly is the ability to target users with flag values that are based on rule configurations. These rules allow you to use attributes that can be configured as part of the user connection to influence the flag values received by the application.

Do you want to ensure that "dev users" are receiving the latest and greatest features or testing out a new component while "prod users" are receiving their existing stable configuration? This concept is one of the hallmarks of LaunchDarkly and core to the idea of "dark launching" features.



The screenshot shows the LaunchDarkly configuration interface for a flag named "ui-refresh". At the top, there's a status bar indicating the flag is "On", evaluated 4 hours ago, and currently serving "true". Below this, the "Target users who match these rules" section is active. It shows a rule where the "userCohort" attribute "is one of" the values "dev-users". The "SERVE" dropdown is set to "true". An "Add rule" button is visible below the rule list. The "Default rule" section shows the "SERVE" dropdown set to "false". At the bottom, there's a note: "If targeting is off, serve false". A "Save changes" button is in the top right corner.

Automating Percentage Rollouts

If you have a new feature you want to roll out slowly to groups of your users, or simply want to ensure the feature can continue to scale without triggering major incidents as it rolls out, a percentage rollout gives you time to adjust.

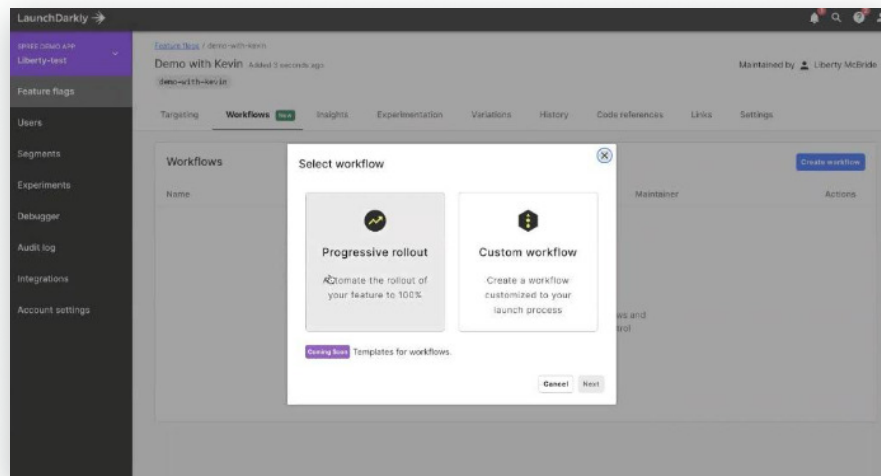


With a percentage rollout, you gradually release a new feature to a certain percentage of your user base—10%, 20%, 40%, etc.—before making it available to everyone. The benefits of this approach allow you to reduce risk, gauge reception, and improve quality.

We've made the automation of percentage rollouts simple in LaunchDarkly. Below, we'll explain the easy way to perform a percentage rollout, and also a more manual way for when you really want to get more hands-on with it.

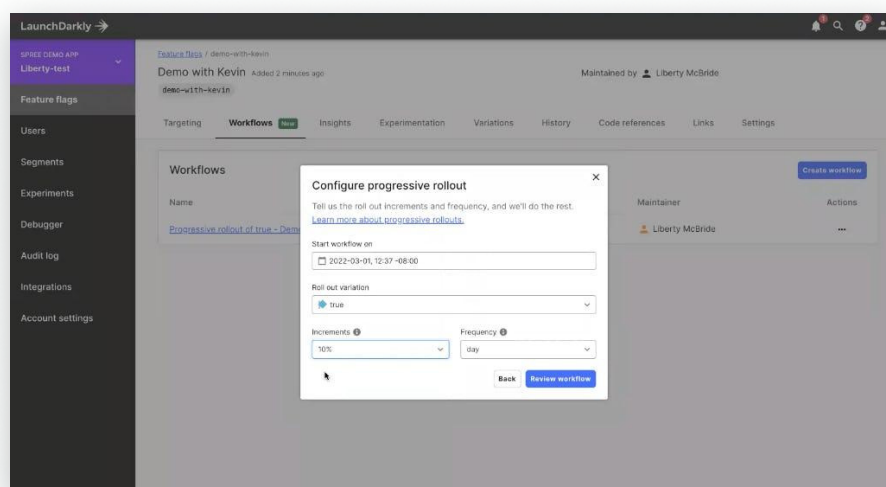


How to automate a percentage rollout in LaunchDarkly



From the LaunchDarkly home interface, navigate to a flag and click on the Workflows tab. When you click "Create workflow," you'll be given the option to either do a progressive rollout or a custom workflow. If you want to do a progressive rollout to all of your users, you just need to select:

- When the rollout should begin
- Which variation you want to roll out
- The percentage increments of the rollout (10%, 20%, etc.)
- The frequency you want the rollout to progress (say 20% every two days)



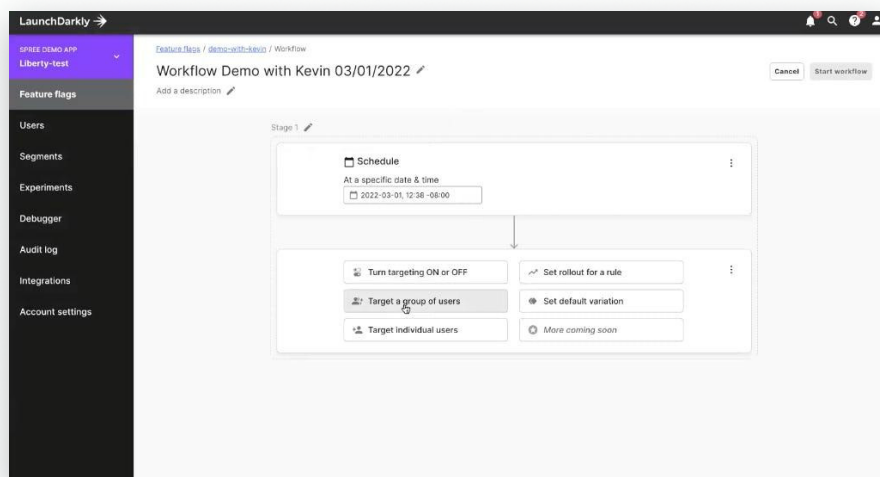
One key thing to note: this method starts your rollout from zero percent of users having received it.

So, if you already have an existing feature partially rolled out to a certain percentage of users (say maybe 40%), and you select the progressive rollout workflow, it will overwrite any previously defined rollouts. Therefore, it's best to select this option when you're starting a rollout from scratch.

Want to preserve the experiences for currently targeted users? This next section will show you more about building a custom workflow.

How to perform a percentage rollout manually in LaunchDarkly

Let's say you want to target a specific set of users ([canary release](#)). In that case, we recommend choosing the custom workflow option. This will open the [Workflow Builder](#), where you can create customized, flexible workflows to fit whatever business processes you follow. You can set up one or multiple approvals, target user segments, and use each stage of the feature release to coordinate on a timeline of your choosing.



A custom workflow allows you to automate a rollout to a specific group of users instead of all users. This can be really helpful during major architectural changes, or with high-risk features.

The screenshot shows the 'Schedule' configuration window in LaunchDarkly. At the top, it says 'Choose to schedule this workflow stage to start on or after a specific date and time or at a later time relative to the workflow start time.' Below this, there are three fields: 'Later' (a dropdown menu), 'After' (a text input with the value '5'), and 'Days' (a dropdown menu). A blue plus icon and a downward arrow indicate a connection to the rule configuration below. The rule configuration is titled 'Rule 1'. It has an 'IF' section with a dropdown 'User is in segment' and a tag 'Beta users' with a close button and a plus icon. Below the 'IF' section is a 'SERVE' section with a dropdown 'a percentage rollout'. Underneath is a horizontal bar chart showing a rollout from 0% to 100%. Below the bar chart, there are two rows: 'true' with a value of '50 %' and 'false' with a value of '50 %'. At the bottom, there is a '+ Advanced' link.

Alternatively, when you click on a flag, you can manually target users you want to receive the rollout and schedule the release for a future date.

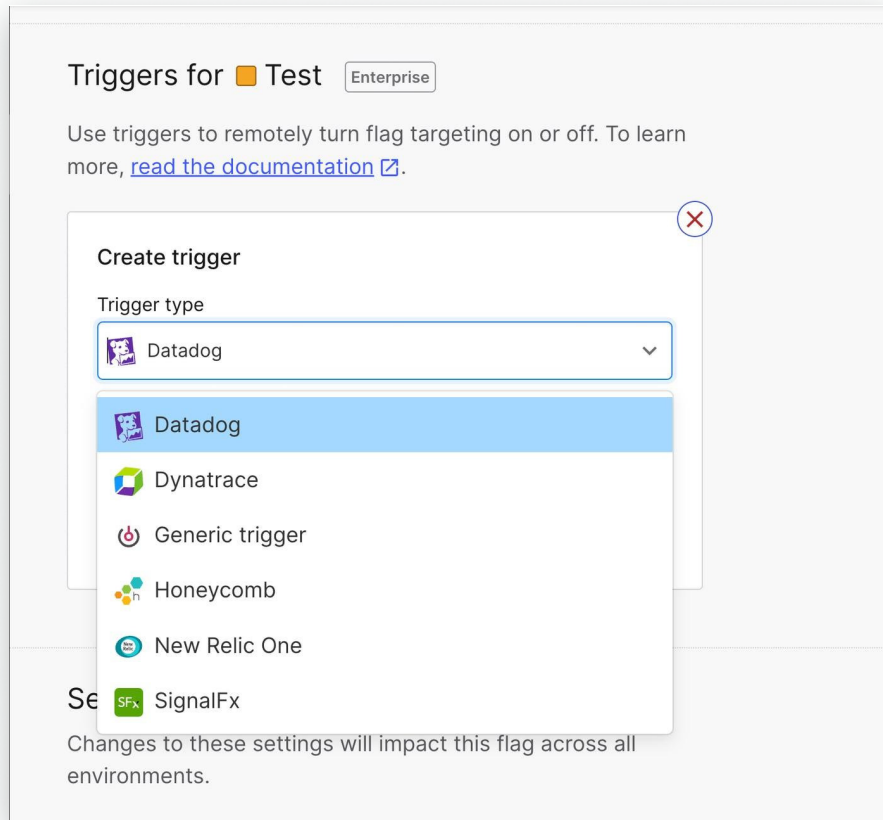
One last note: soon, we'll be releasing workflow templates that allow you to build a workflow, save it, and use it again at your discretion, so stay tuned. Until then, feel free to dive deeper into this topic [in our Docs section on the LaunchDarkly site](#).



02 Flag Triggers

Flag triggers give you the ability to integrate feature flag changes with different platforms. Out-of-the-box integrations include flag triggers for many observability platforms such as [DataDog](#), [New Relic](#), [Dynatrace](#), and [Honeycomb](#). Integrating flag triggers in these platforms allows you to trigger flag changes from within dashboarding tools and programmatically based on metric values and events within those platforms.






You can also create generic flag triggers that provide a URL to enable or disable specific feature flags. This URL can be tied into API requests as part of your development pipeline, or even provided to developers on teams to self-manage features without being forced to log into the user interface.

These URLs can be called on demand to enable or disable the targeting rules that manage features. Problem with a deployment late on a Friday evening after a last-minute feature release? Simply call the disable URL to turn the feature off.

This is an often overlooked but extremely powerful capability for integrating LaunchDarkly into other platforms. [Learn more about Flag Triggers in our documentation.](#)



Triggers for Test Enterprise

Use triggers to remotely turn flag targeting on or off. To learn more, [read the documentation](#) .


Generic trigger ...

Trigger updates flag targeting to ON



<https://app.launchdarkly.com/webhook/trigge> 



Copy and save this URL now. The URL will be obscured after you leave the page. To learn more, [read the documentation](#) .

Create trigger ✕

Trigger type



Generic trigger ▼

Action

Update flag targeting to OFF ▼

Save Trigger

Flag prerequisites

Flag prerequisites allow you to configure dependencies on other feature flag values in order to enable another feature. For example, you might develop a "new login" component as part of your "new UI" refresh, both of which are controlled by individual feature flags. It wouldn't make sense to roll out the "new login" feature flag before the "new UI," so you set the "new login" flag to require a "true" result from the "new UI" feature flag. This helps protect your user experience and ensure that features are being released in your intended order.



The screenshot shows the 'Targeting' section of a LaunchDarkly interface. At the top right, there is a 'Save changes' button. Below the title, a green toggle switch is labeled 'On'. To its right, three status indicators are listed: 'Evaluated 5 hours ago' (with a clock icon), 'Currently serving true' (with a green diamond icon), and 'On save, will serve true' (with a green diamond icon). The 'Prerequisites' section is expanded, showing a list of prerequisites. The first prerequisite is 'Updated Header' with an 'ON' toggle. To its right, a dropdown menu is open, showing 'true' (selected) and 'false' (unselected). Below the prerequisites list, there is an 'Add prerequisite' button. At the bottom, there is a section for 'Target individual users'.

Using flag prerequisites to plot out the path of features you want to enable as a multi-part release allows you to truly orchestrate the rollout of a new set of capabilities across your application while protecting yourself from the bad experience of rolling out a feature before the rest of your platform is ready for it. These flag prerequisites can also leverage targeting rules to ensure that the right users are receiving the right features. Check out more examples in our [flag prerequisites docs](#).



03 Relay Proxy at LaunchDarkly

Have you ever wondered how to use the [Relay Proxy](#)—or even what it is in the first place?

You're not alone. We'll go over what the Relay Proxy is and what it does below. However, it's difficult to be prescriptive when it comes to leveraging Relay Proxy, because each team's needs and usage/deployment patterns are unique. So, we'll tell you exactly how we at LaunchDarkly use the Relay Proxy. Hopefully, that can provide guidance and some inspiration for how you could also get the most from it.

Note: If you're looking to dive straight into the Relay Proxy, we have [plenty of documentation](#) that will walk you through different use cases, configuration concerns, cost structures, best practices, and more, so feel free to check those out.



What is the Relay Proxy?

The [Relay Proxy](#) is an open source project supported by LaunchDarkly that enables multiple servers to connect to a local stream rather than making several outbound connections to our streaming service.

Each of your servers connects to the Relay Proxy only, which maintains the connection to LaunchDarkly. You can configure the Relay Proxy to carry multiple environment streams from multiple projects. Some common use cases for the Relay Proxy:

- Reducing your app's outbound connections
- Keeping user data private
- Facilitating faster connections
- Meeting continuation of service requirements
- Reducing firewall configuration complexity for your customers
- Increasing startup speed for serverless functions
- Reducing operational work when creating new projects and environments

How we use the Relay Proxy at LaunchDarkly

When most folks use the commercial version of LaunchDarkly with the Relay Proxy, they use it to talk to the main commercial instance. Internally, we use it to sit in front of what we've dubbed our "catfood" instance. (Catfooding is like [dogfooding](#), but since our parent company is [Catamorphic](#), we call it catfooding.)

This relay sits in front of our private instance—it's very specific to how we use it. We have many different services that need to connect to our internal catfood instance to pool flag information.



Put simply, the Relay Proxy sits in front of the catfood instance so that the other services can talk to the relay instead of talking directly to their internal catfood instance. The relay acts as a sort of caching mechanism and backup. It allows the catfood service to break without breaking all the new and existing clients trying to read information from it.

Deploying the Relay Proxy with Spinnaker

We deploy like we do all of our other applications: with Spinnaker. This is an app service built at Netflix that the company uses to deploy their own stuff as well. We use a version provided by armory.io. You can essentially give Spinnaker a recipe on how to deploy something, and it does it automatically. We love it.

Deployment is done in AWS. Spinnaker deploys EC2 quite well, and since we generally use EC2, we use EC2 instances to house the LaunchDarkly Relay Proxy nodes. We then tell Spinnaker which regions in AWS to deploy and how many nodes, and the service handles the rest.

Automating upgrades with the Relay Proxy

Upgrades are easy. We've set up our pipeline so we can simply tell it which version of the Relay Proxy to deploy, hit a button, and it automatically makes upgrades everywhere. We run the relay in five regions, so it's a highly available service that's capable of withstanding outages. Everything can continue working properly even if all but one region is down at any given time.

We hope this helps with your consideration or usage of Relay Proxy. Again, if you want to learn the full ins and outs of the Relay Proxy, please [check out our docs](#).



Learn more tips and tricks

We hope you've found these tips helpful. If you're interested in staying updated on the latest tips and best practices, be sure to check up on the [LaunchDarkly blog](#), where we're always adding new posts on how to get the most out of our platform.

LaunchDarkly offers a [feature management platform](#) designed for the entire organization. We believe the effective use of feature flags is an organization-wide endeavor, not something limited to a few developers. All teams across the organization can benefit from the use of feature flags. LaunchDarkly grows with you as your feature flagging practice expands from a single flag deployed by a single developer, to team-wide use, to delegating control of flags to other teams to meet their use cases.

To see for yourself, [request a demo](#) of the LaunchDarkly feature management platform or [start a free trial](#).

Request a demo



LaunchDarkly ➔

launchdarkly.com