eBook

Mitigate your Infrastructure Migration Risk with Feature Flags

CloudBees

Table of Contents

Introduction

Cloud migration is a necessary yet risky endeavor that can disrupt services and introduce unexpected costs. This is backed by a 2024 Palo Alto study where 50% of respondents stated if they had a redo in their cloud migration, they would spend more time refactoring their applications instead of migrating with minimal change. A key driving factor is the total cost of the ownership, where 30% of cloud costs go towards overhauling legacy apps.

Mastering cloud migration calls for advanced techniques that dovetail with existing continuous integration and continuous delivery (CI/CD) methods. Progressive delivery and feature flags are fast-evolving techniques that can keep migration projects focused while sustaining momentum.

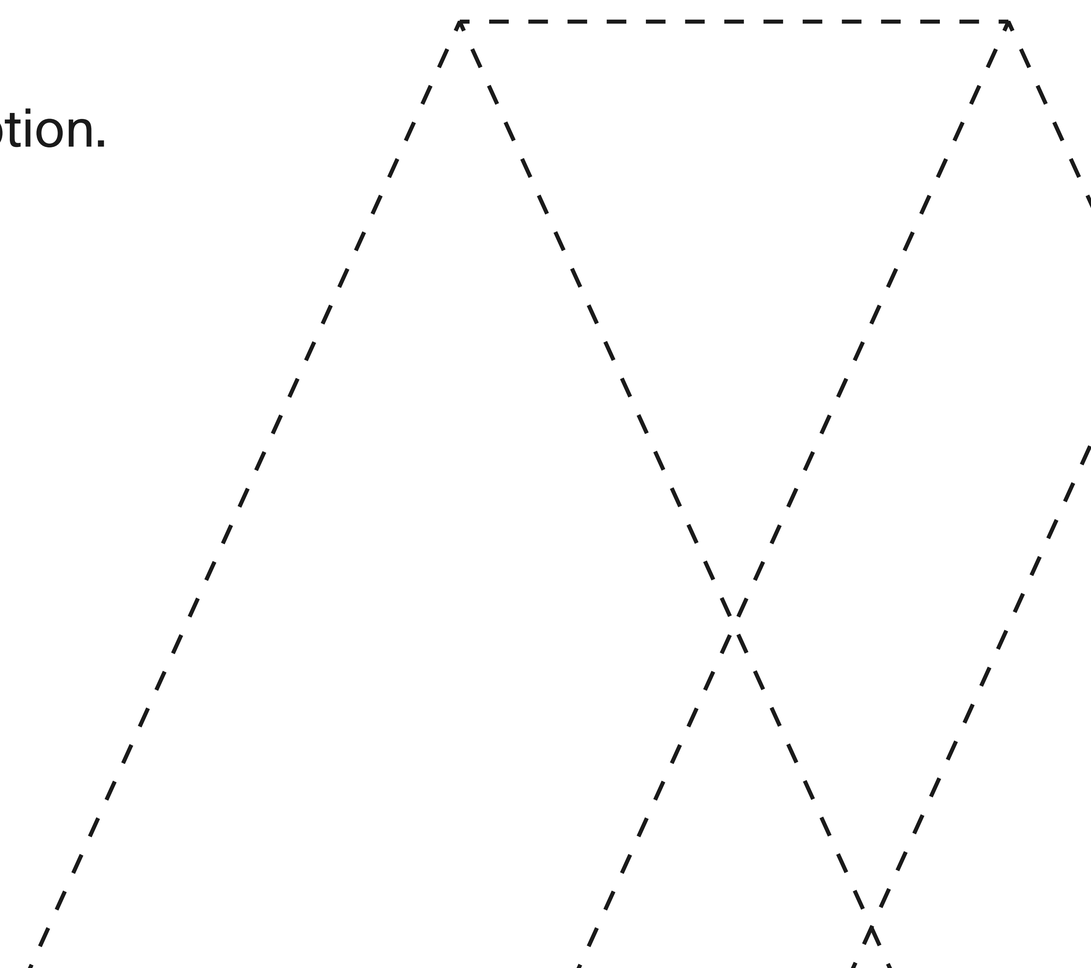
The Challenges of Infrastructure Migration

By migrating infrastructure to the cloud, digital enterprises can realize some notable benefits. Scale-out cloud architectures enable applications to rapidly scale up transaction volumes while maintaining performance. Cloud architectures also promise more flexibility and agility as part of a DevOps discipline that supports fast deployment.

However, these benefits don’t come without risks. Migrating to new infrastructure risks downtime if the code and infrastructure aren’t compatible. It could break data integrity if transactions fail after the big switchover, creating a mess that requires complex recovery and affects longer-term operations.

Even if the migration failure isn’t cataclysmic, performance or usability issues could create problems of their own. Bewildered users could clog up the help desk with calls, creating bottlenecks and user acceptance issues.

Each of these issues takes a toll on your brand’s perception.



A New Approach to Cloud Transformation

Developers often test for performance and reliability issues before deployment, but they're a best-effort attempt. Testing and staging environments might not accurately reflect production platforms.

Typical monolithic code migrations involve running legacy and new systems side by side for a period until the project team is certain that the new system works properly. Then, they turn the old system off. That's a frightening prospect when handling code bases millions of lines long—especially if migrating infrastructure to the cloud changes its underlying architecture.

Progressive delivery takes a new approach to code deployment by testing it safely in production. This technique doesn't abandon traditional testing, but it swaps monolithic big-bang code roll-outs in favor of gradual feature deployments. It gives you the luxury of testing your code against real live production data, leading to more reliable code with proven performance on cloud infrastructure.

Progressive delivery tests new features in production while simultaneously using those features on legacy systems. The development team only switches over an individual feature permanently when it has proven itself in the production environment.

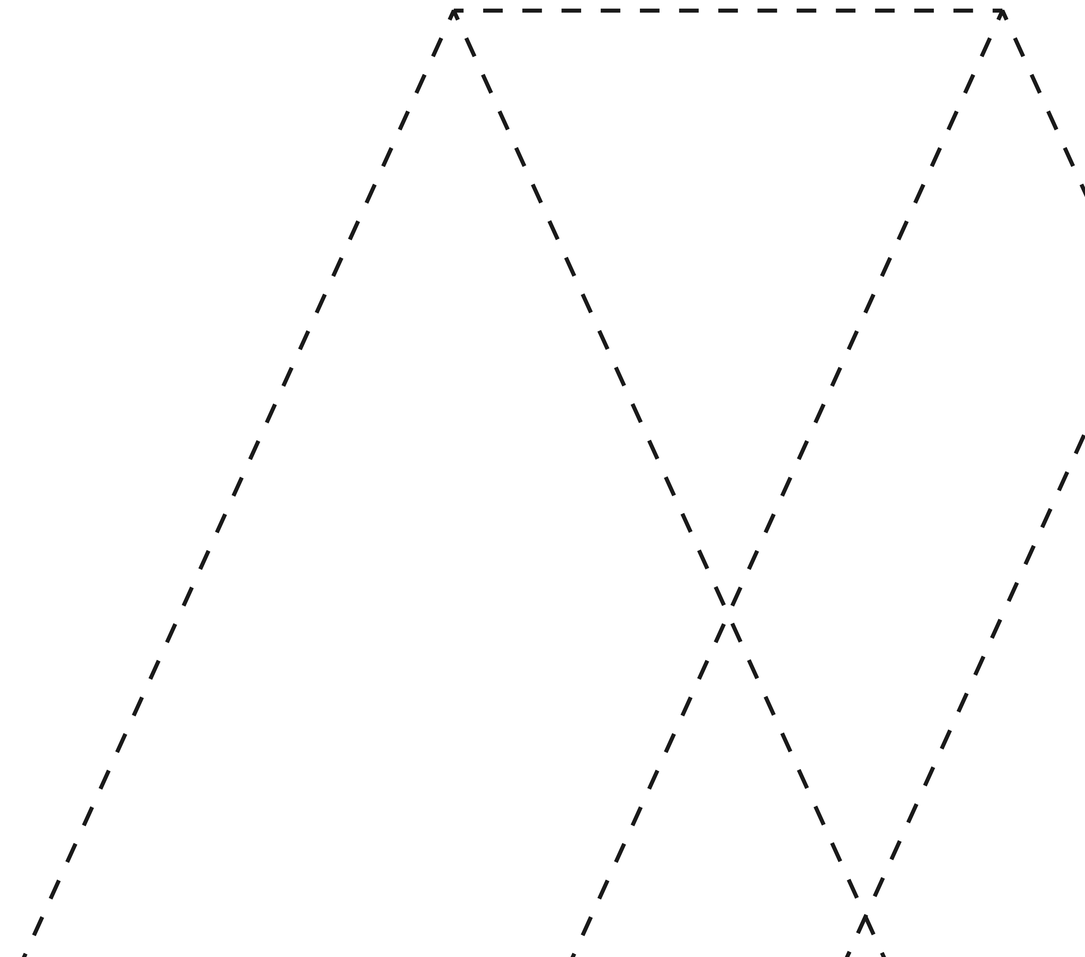
> Progressive delivery tests new features in production while simultaneously using those features on legacy systems.

How Feature Flags can help Mitigate Risk and Add Value

Managing this gradual migration of individual features to new infrastructure involves adapting something over which you have complete control: your code. You can insert control switches into your code that define if that code runs, where, and for whom. This is where feature flags come in.

Feature flags are variables in your code that let you control software functions at any level, from a database read to a form element in a browser. By flicking these variables on and off, developers can change application behavior in production without having to rebuild and redeploy.

Developers can make feature flags as granular as they like. This enables them to change functions at a minute level, including those that aren't user-facing. Their fast, granular flexibility makes them valuable assets when migrating infrastructure to the cloud.



Here are four powerful benefits that feature flags can deliver to project teams during migrations:

Fast testing and rollback

Feature flags enable developers to test specific services against cloud infrastructure so they can garner more precise insights into a feature's performance in the new environment. Their ability to change software functions without redeployment makes it easy to quickly step back if things go wrong.

This ability to roll back deployments lets teams gradually test new infrastructure, like walking gingerly on ice. This strategy, called "expand-contract," is useful when handling tricky migration tasks.

One example of a feature flag-based expand-contract migration pattern is migration between two databases or schemas. Project teams can use this technique to maintain data integrity in the original data store while testing production interactions with the second one. Feature flags allow developers to test each database at the same time rather than taking a leap of faith and cutting over between them. They can do this by creating separate read and write flags for each database that let them test their code's performance against the new infrastructure.

Developers can write to the old and new databases simultaneously by turning on both write flags. When the code proves its reliability and performance writing against the new database, they can test its ability to read from the new schema accurately. They can simply set a flag to turn on reads from the new database, logging any differences.

This approach enables the development team to roll back its database migration at any point because it keeps the original data store intact. If it all goes horribly wrong, users need never know the difference.

Feature flags go beyond activating and deactivating software functionality. Developers can also use them to turn on a feature gradually for different user groups. That enables them to ramp up volume over time, testing out more production data and spotting any early issues along the way.

A smooth shift to cloud-native infrastructure

Database migrations are just one task facing project teams migrating to cloud-native applications. Migrating infrastructure to the cloud often involves rearchitecting monolithic code bases to microservices. Developers can use feature flags to handle this task by supporting a migration pattern known as the strangler.

The strangler pattern takes its name from the strangler fig tree. This plucky little plant grows on a host tree and slowly chokes it to death, eventually replacing it entirely.

In cloud-native migrations, the new code base is the fig tree and the old code is the host. The new code grows over time to replace the legacy codebase. It leaves the incompatible legacy code for lift-and-shift operations or simple decommissioning.

Developers can use feature flags to gradually switch legacy functions to the new code base while closely evaluating their performance. This enables them to replace the old code base in functional chunks that make sense to the business.

Easy identification of user issues

Using feature flags to gradually switch across different user groups to cloud-native infrastructure is also a great complement to user acceptance testing (UAT). Project teams will hopefully have conducted UAT earlier in the development process, but it’s an inaccurate science. Users’ needs change, and platform performance issues can also make what seemed fine in UAT unworkable in production.

Supporting canary releases with feature flags enables the project team to roll back deployment if the folks in accounting or customer support suddenly identify a show-stopping performance or usability issue.

Increased leadership confidence

Migrating code using feature flags helps to prove the technical performance of new code and infrastructure. It can also help to improve business managers' confidence in what's happening by allowing them to assess the migration against their own key performance indicators.

Cloud sticker shock is a serious problem for companies that don't properly monitor their expenditure, and yet only 15% of organizations have a mature and evolving FinOps practice, per The FinOps Foundation. Feature flags can test the financial performance of individual features against live production data on new infrastructure. That's a valuable source of FinOps data.

3 Feature Flag Best Practices to Consider

1 Integrate feature flags tightly with your CI/CD pipeline

Ensure that your CI/CD tools recognize feature flags and use them as data sources. This involves two-way communication between the other tools in a CI/CD pipeline and feature management software. For example, configure your tools to see feature flag audit logs so that you can easily see who switched on a flag and when.

CI/CD tools should also be able to configure feature flags themselves. Changing user permissions and targeted feature flags makes it easier to control the use of flags in software delivery, ensuring that feature deployments align with business goals.

2 Learn in production

One of the biggest benefits of DevOps is its potential to gather performance and usage data in the field. Supercharge this capability by using feature flags to learn from your production deployments.

Individual feature deployments offer useful data based on feature usage among different user groups, helping you to understand their popularity. This can inform an effective release strategy, enabling you to roll back and tweak features as necessary.

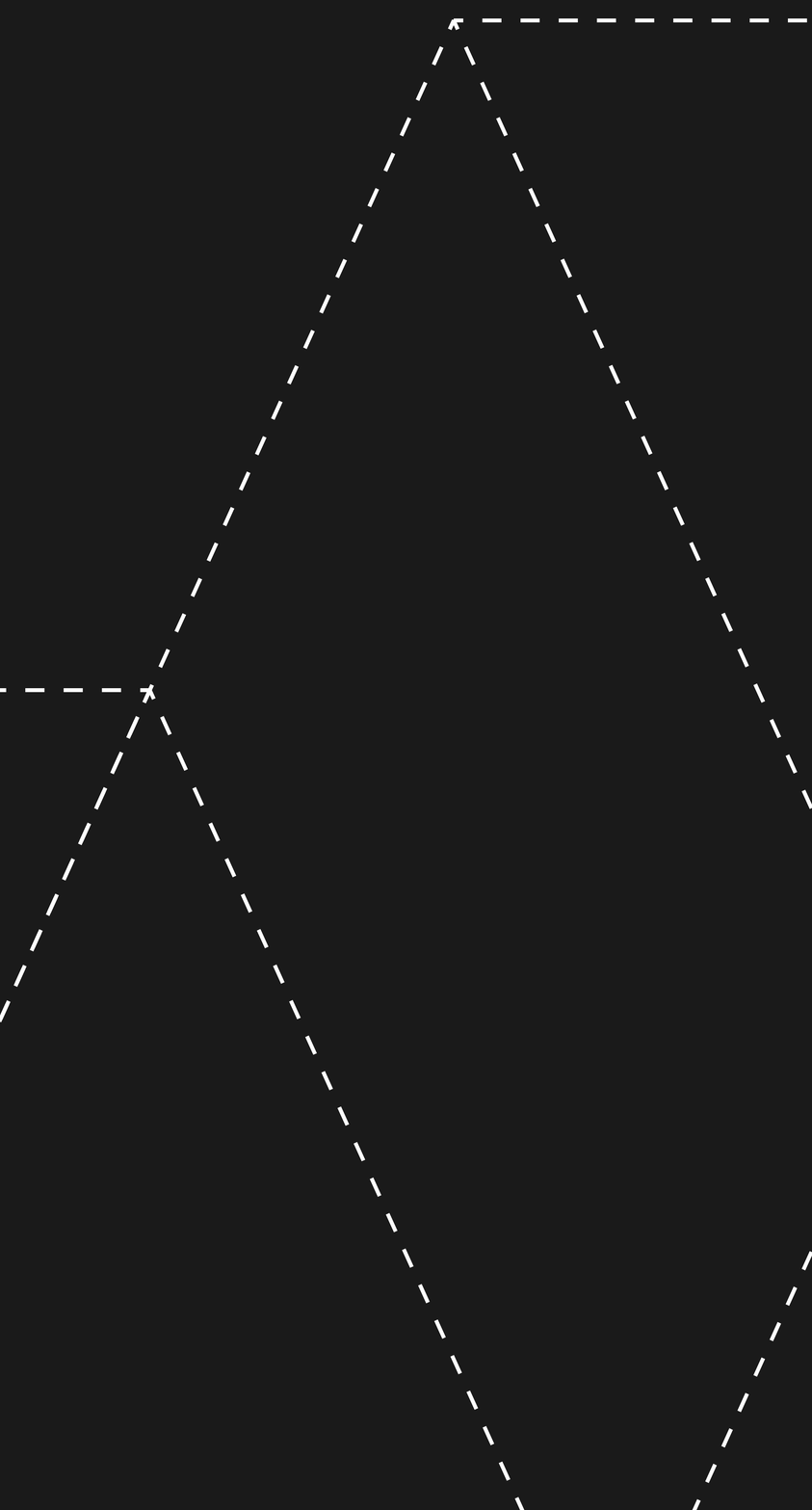
3

Manage your entire lifecycle

Feature flags are like any other digital asset, such as sensitive enterprise data or virtual machines. They can be useful, but unless you manage their lifecycle completely, from cradle to grave, they can become toxic.

A feature flag that gets forgotten can quickly become a source of technical debt, creating more complexity in your architecture and code. Plan milestones for your feature flags, including retiring them to keep your code clean. That means formalizing it in your code management process and making someone responsible for it. Include hooks between your code and CI/CD tools so that you can easily see which flags are in use at any time.

Avoid the temptation to manage feature flags in a simple spreadsheet or text file. A dedicated feature management system will help you bind your flags tightly into your DevOps workflow so that you can reap the full benefits of this powerful development technique.



The Business Value of Feature Flags when Migrating Infrastructure to the Cloud

Feature flags and progressive delivery are powerful allies during cloud and infrastructure migrations, allowing development teams to test changes safely, reduce risk, and gather real-time feedback in production. By targeting specific user groups and rolling out features gradually, teams can validate performance in new environments without slowing innovation.

CloudBees Feature Management, as part of the CloudBees Unify solution, empowers organizations to orchestrate these migrations with confidence. Feature-level controls, coupled with full visibility and governance across the pipeline, ensure you can ship safely, even in the midst of large-scale architectural changes. And if something goes wrong? Rollback is immediate and damage-free.

With CloudBees Unify, teams don’t just migrate faster, they migrate smarter. By embedding feature flags into your CI/CD pipeline and connecting them to the broader DevOps ecosystem, you reduce risk, improve developer productivity, and accelerate time-to-value for your cloud-native initiatives.

Learn more

The ROI of Using Feature Flags in Software Delivery

Download the eBook
[5 Things You Need to Get Started with Enterprise Progressive Delivery]

Download the eBook
[Feature Flags: Should I Build or Buy?]

Read the White Paper
[Feature Flags Across CI/CD]

Get a Free Trial today! cloudbees.com/capabilities/feature-management

CloudBees, Inc.
cloudbees.com
info@cloudbees.com