eBook

5 Things You Need to Get Started with Enterprise Progressive Delivery

CloudBees

Table of Contents

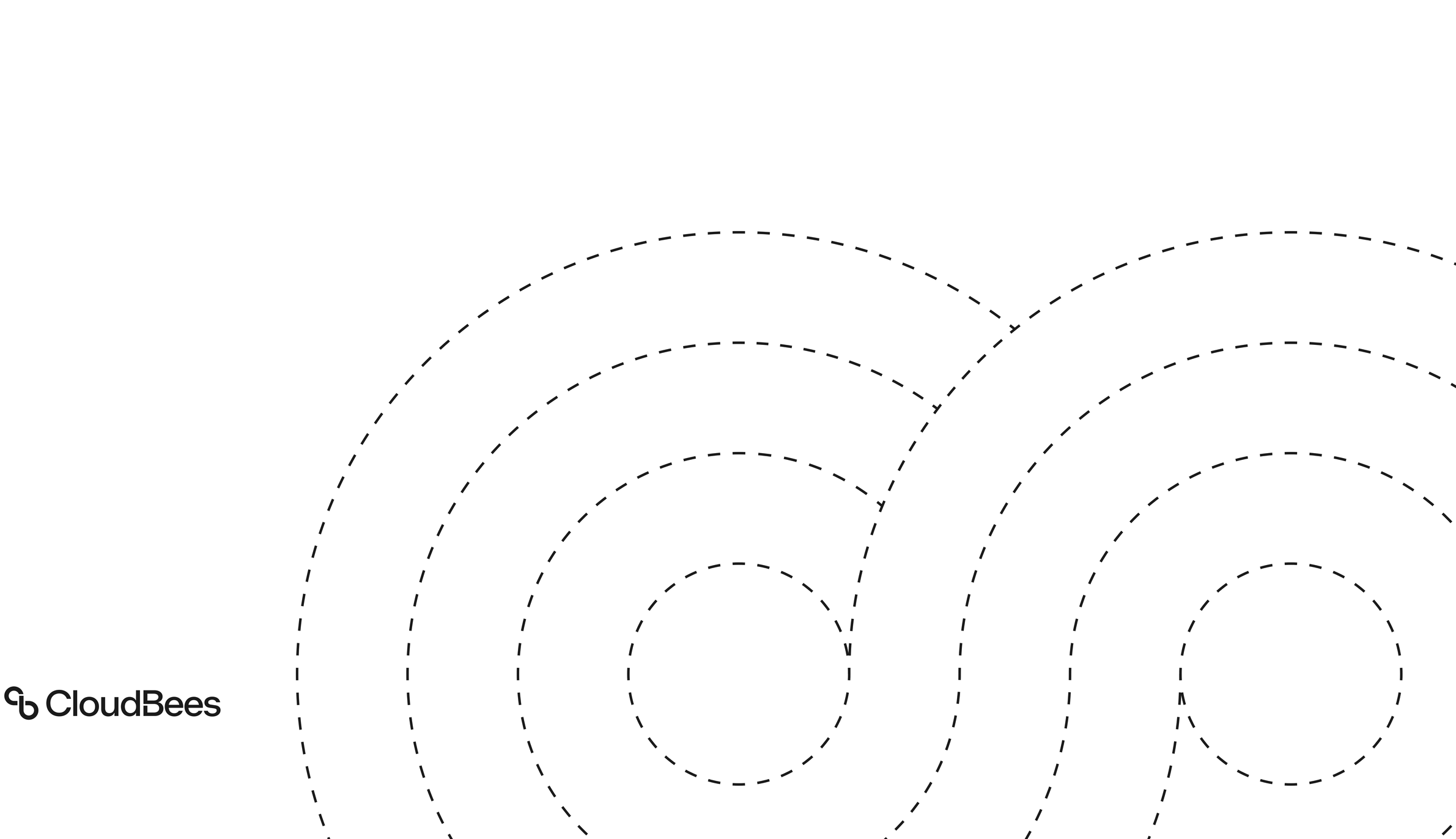
Introduction

When you've implemented CI/CD practices and tooling in your organization and the concept of testing in production using feature flags is no longer a scary concept, you've reached a significant milestone. You've invested in highly automated testing and, most likely, your pipelines and necessary culture shifts. You're seeing those investments pay off in higher velocity and higher quality.

So, if your organization is already comfortable with shifting testing left in the software delivery lifecycle, what's the next step in your DevOps journey? Enter enterprise progressive delivery.

Enterprise progressive delivery is a continuation and natural evolution of the concepts of continuous integration (CI) and continuous delivery (CD). In progressive delivery, developers roll out features incrementally, first to a small, pre-defined audience (such as internal QA teams) and eventually to larger audiences in a controlled, measured manner.

To implement progressive delivery at enterprise scale, governance, scalability and integration with other CI/CD tools are critical. It's a repeatable, automated way to test and learn in production using real behavior and feedback from users. Like canarying, A/B testing or blue-green deployment, the end goal is to ensure that higher-quality code reaches users fast. To create a repeatable progressive delivery practice at enterprise scale, there are a few key considerations to keep in mind. This eBook serves as a guide to starting your enterprise progressive delivery journey.



Who Is Enterprise Progressive Delivery For?

Before we dive in, you may be wondering—who is enterprise progressive delivery for? Any team interested in improving their DevOps performance can benefit from these practices.

Where Does Your Organization Fit In?

Baseline CI Practices	Continuous integration (CI) is one of the core best practices of agile development. Developers should be merging changes as often as possible into the main branch.
Baseline CD Practices	You have an automated method for continuously delivering new releases to production at scale.
Feature Flags	Feature flag capabilities are important to enterprise progressive delivery. Feature flags allow fine-grained control at the feature level, which is essential to progressive delivery practices.

The 5 Things

1 Common Visibility of Feature Flags

Your CI/CD pipeline should be able to receive feature flags as input, and it should be aware of the gated features in each release according to the environment. It should be able to present the configuration of your feature flags as well as current release criteria.

Why Does This Matter?

Feature flags and CI/CD are better together. In fact, you could say that feature flags are an essential building block of enterprise progressive delivery. You can use feature flags to test your CI/CD toolchain, perform experiments and understand what's working and what's failing—at the feature level and otherwise. Remember: this practice reaches enterprise scale only if all teams can view feature flag information across the software delivery lifecycle.

2 Common Controls and Governance

Your CI/CD pipeline should be able to change the value of individual features, update feature flag targeting, control feature flag change permissions (such as user permissions and checks before advancing to the next state of the pipeline) and ingest feature flag audit logs (to understand who opened particular features, and when).

Why Does This Matter?

Common controls and governance help you work efficiently at scale. You will need to avoid bottlenecks, maximize developer time and ensure that you meet enterprise release protocols and standards across teams.

3 Smart Automation

CD typically has a tactical focus, but to introduce more modern DevOps practices, your team may find itself concentrating on feature value and strategy. Your CD pipeline should be able to ingest performance data connected to feature flags, produce feature release templates and define consistent rollout and rollback policies.

Why Does This Matter?

Shifting focus from from a release level to feature level means that many more features will need to be tracked. Smart automation is essential for releasing your features without a lot of manual effort each time, allowing for the benefits of feature-level release granularity without extra release overhead.

4 Process Shift at the Team Level

You'll need to establish feedback and revision processes for features being tested in production, and determine who is responsible for releasing features (such as release management, developers, EMs or PMs). Feature flags must also be managed before they turn into technical debt. Most have a built-in "expiration date," and they should be actively monitored and retired at the right time.

Why Does This Matter?

Robust processes provide a guard rail to help establish and grow enterprise progressive delivery practices in your organization. Everyone needs to buy in to process changes to ensure that teams are moving in lockstep and there is no disconnect.

5 Cultural Shift at the Organizational Level

Enterprise progressive delivery involves as much of a cultural transition as a technological one. Your organization will need to get comfortable with releasing "incomplete" code, and you may even need to change policies on how code can be pushed to production. While some best practices may need to be adjusted, enterprise progressive delivery is ultimately a shift in cultural mindset—from the delivery of groups of features within releases to the delivery of value at the feature level. The philosophy of how features are created, released and measured within an organization will transform.

Why Does This Matter?

Enterprise progressive delivery involves a reshaping of the organizational mindset, which can make buy-in difficult until progressive delivery and the safeguards in place for it are well-understood. This cultural shift will be ongoing, but stick with it. Arriving at a repeatable, automated means of understanding and adjusting to real user data in production can carry your organization to new levels of efficiency.

Summary

Enterprise progressive delivery is a powerful evolution of DevOps practices, shifting the focus from the release level to the feature level to improve speed, safety, and customer impact. While it builds on the principles of continuous delivery, it introduces a more granular approach to risk mitigation and innovation through real-time control over feature exposure.

With CloudBees Feature Management, powered by the open and extensible CloudBees Unify solution, enterprises can adopt progressive delivery at scale, confidently managing feature rollouts, automating governance, and integrating seamlessly with the broader software delivery toolchain. This shift isn't just about new tools; it's about evolving your culture, gaining deep visibility into what's running in production, and making smarter decisions faster.

Get started for free! cloudbees.com/capabilities/feature-management

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