eBook

The Release Gap: Why CI/CD Isn’t Enough

CloudBees

Introduction

Most organizations believe they already have what they need to release software: CI/CD pipelines, automated tests, and deployment tooling. But the lived reality tells a different story:

- Releases feel slow, fragile, and unpredictable
- Teams struggle to answer basic questions about what is shipping
- Incidents take too long to triage
- Audits are painful and manual
- Environments constantly drift out of sync

This is not because teams lack automation. It’s because they’re trying to release a distributed system with tools designed for a monolith.

The core truth:

> CI/CD automates the steps.
> Modern releases require orchestrating the system.

And the gap between automating pipelines and coordinating a distributed application is exactly where today’s enterprises struggle.

Why This Matters

When organizations rely on CI/CD alone to coordinate modern releases, they experience growing business risk:

- Higher change failure rates due to environment drift
- Operational costs from manual coordination between teams
- Slow incident recovery because no one knows what changed
- Compliance exposure from scattered approvals and missing evidence
- Customer trust erosion from production defects that never appeared elsewhere

These challenges compound as systems scale, architectures become more distributed, and AI accelerates the rate of change across repositories.

Most leaders feel it as a growing sense of release anxiety.

Where CI/CD Stops — And Release Complexity Begins

CI/CD pipelines answer operational questions like:

- Has this component passed its tests?
- Can we deploy this artifact?

But releases demand a different set of questions:

- What exactly is in this release?
- Which components were validated together?
- Where is each version running right now?
- Who approved this? Where’s the evidence?
- If something breaks, what is our known-good rollback state?

These are not CI/CD questions — they are coordination questions, and CI/CD has no native concept of a release.

This misunderstanding is why many organizations think they “already do release orchestration” when in reality, they have automated pipelines... but no system that orchestrates change across the whole application.

The Common Failure Patterns

Across thousands of engineering teams, the same patterns appear:

1. **Fragmented understanding of what a release actually is**
   Teams know their own services, but no one sees the whole system. This leads to competing truths about production.

2. **Environment drift invalidates testing**
   QA ≠ staging ≠ production.
   Tests pass in combinations that never ship.

3. **Evidence and approvals live in scattered tools**
   Slack, email, Jira, spreadsheets, CI logs → audit chaos.

4. **Slow incident response due to poor traceability**
   When you can’t answer “what changed?” quickly, MTTR explodes.

5. **Coordination overhead grows faster than the org**
   More teams, more services, more pipelines → more manual governance.

6. **A stitched-together toolchain masquerades as a release process**
   Organizations attempt to approximate release management using Jenkins, GitHub Actions, Argo CD, Slack approvals, and spreadsheets. The result isn't a release process—it’s a collection of tools pretending to be one.

Why These Problems Are Getting Worse, Not Better

Three industry shifts amplify release complexity:

Distributed Architecture Is Now the Default

Microservices, shared libraries, data pipelines, external APIs, managed services—systems are more interconnected than ever.

Failures surface at the intersections between services—not in isolation.

AI Accelerates Change Beyond Human Coordination Capacity

AI increases the volume and velocity of code changes across repositories. Velocity without orchestration increases risk.

Regulation and Compliance Expectations Are Rising

Auditors expect real-time traceability and consistent processes. CI/CD alone cannot provide that.

Practical Self-Assessment: Are You Doing Release Management Without a Release System?

If you answer “yes” to several of these, you’ve outgrown CI/CD alone:

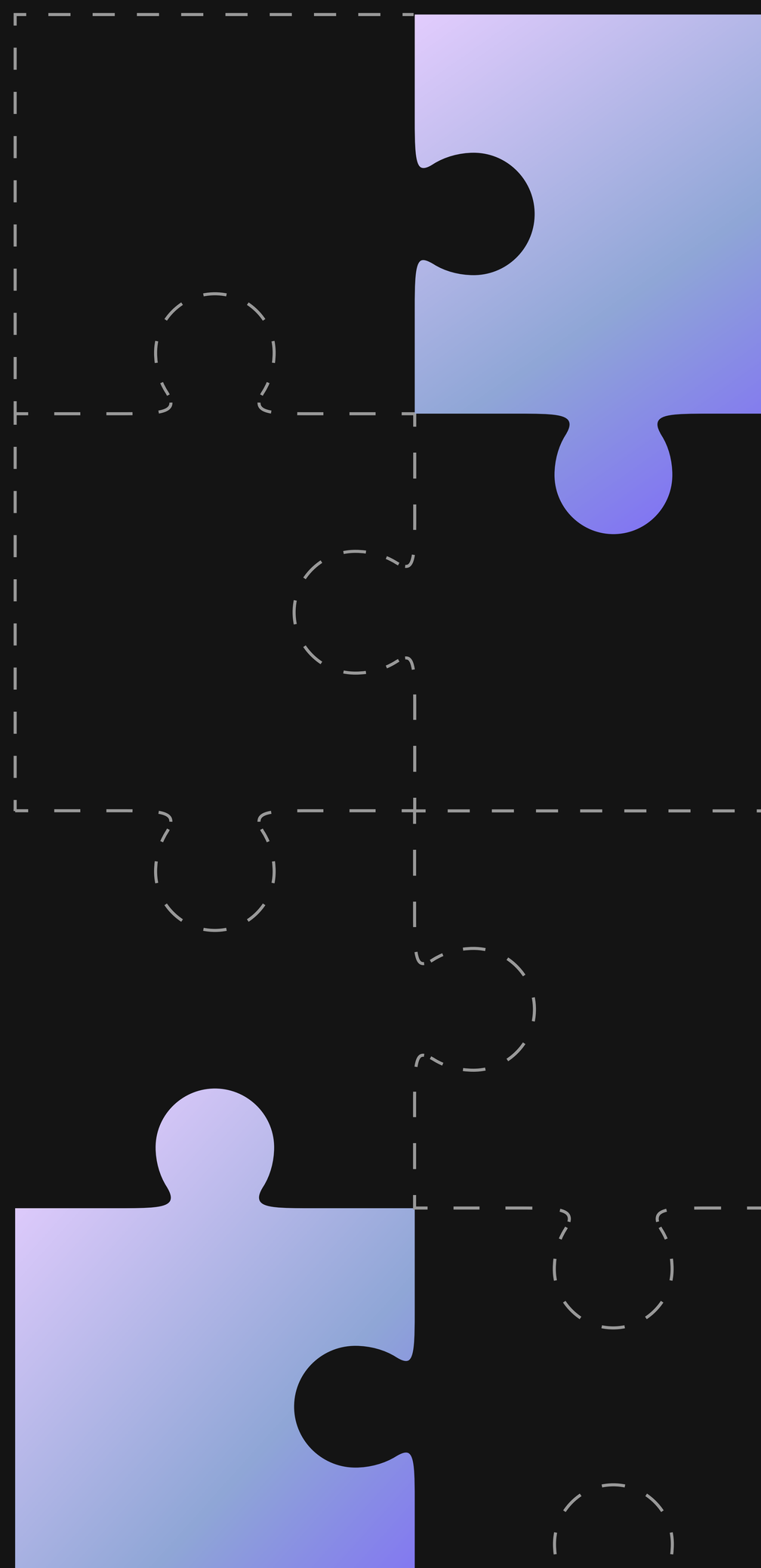
Visibility & Control Gaps	☐ “We can’t quickly answer what’s in this release.” ☐ “Approvals happen in Slack or email.” ☐ “Our evidence isn’t centralized.”
Environment Drift	☐ “QA, staging, and production rarely match.” ☐ “Bugs appear only in production.”
Operational Load	☐ “Releases require too much manual coordination.” ☐ “Rollbacks take hours.”
Scaling Constraints	☐ “The number of services has exceeded our ability to coordinate them.” ☐ “AI is increasing change velocity faster than our processes can absorb.”
Compliance Pressure	☐ “Evidence collection is manual.” ☐ “Audits expose inconsistencies.”

If this feels familiar, CI/CD is doing its job — but your release process is still missing its system of record.

The Capabilities CI/CD Cannot Provide

CI/CD is essential but incomplete.
Pipelines cannot:

- Maintain environment consistency
- Coordinate versioning across services
- Centralize approvals and evidence
- Guarantee fast, predictable rollback states
- Provide visibility across teams
- Define a single version of “ready to release”



Organizations often respond by adding more pipelines, more steps, more automation.
Ironically, this often exposes the gap more sharply.

This is the gap Release Orchestration fills.

CloudBees

What Modern Release Orchestration Provides

Modern Release Orchestration (RO) introduces structure where CI/CD reaches its boundary:

1. A Unified Definition of a Release	A single, authoritative object containing: • component versions/artifacts • evidence • approvals • metadata • environments Everyone sees the same truth. No tribal knowledge required.
2. Payload-Based Promotion	Move the same validated payload through QA → staging → production. This eliminates environment drift — the #1 cause of escaped defects.
3. Centralized Governance & Evidence	Approvals, checks, security scans, and testing are tied to the release itself, not scattered across Jira, Slack, CI logs, and spreadsheets. Compliance becomes continuous. Audits become trivial.
4. Predictable Rollbacks	A coherent definition of the release makes rollback instant and reliable — not an investigation.
5. Cross-Team Alignment	SRE, QA, Security, Release, Platform, and Product all operate from one source of truth.

How High-Maturity Teams Evolve Release Practice

A consistent progression emerges:

1. Model the application — define what ships together
2. Introduce release manifests
3. Adopt payload-based promotion
4. Centralize evidence and approvals
5. Implement release observability
6. Expand governance across teams

The result is more predictable releases and dramatically reduced cognitive load.

The Payoff: Predictability, Confidence, and Calm

Organizations that adopt Release Orchestration report:

- Higher release confidence
- Reduced environment drift
- Faster incident recovery
- Fewer escaped defects
- Lower compliance burden
- Stronger cross-team alignment

But the deeper payoff is emotional:

> Releases stop feeling like firefighting.
> Fridays stop feeling dangerous.
> Teams finally exhale.

Conclusion

Modern software delivery has outgrown the capabilities of pipeline-centric tooling. CI/CD remains essential for building, testing, and deploying individual components, but it cannot coordinate multi-team, multi-service releases or provide the governance and traceability required in today’s distributed environments.

Release Orchestration establishes the control layer that enterprises have been missing. By defining releases explicitly, promoting consistent payloads across environments, unifying approvals and evidence, and enabling clear visibility into what is running where, RO provides the operational discipline needed to deliver software predictably and safely —without forcing teams to abandon their existing CI/CD tools.

Release Orchestration doesn’t replace CI/CD; it completes it by delivering the governance, consistency, and system-level visibility modern delivery demands.

To understand how this model is implemented in practice, discover how CloudBees Unify Release Orchestration simplifies releases across teams, tools, and environments, enabling enterprises to modernize release governance without disrupting how teams build.

Learn more cloudbees.com/capabilities/release-orchestration

CloudBees

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