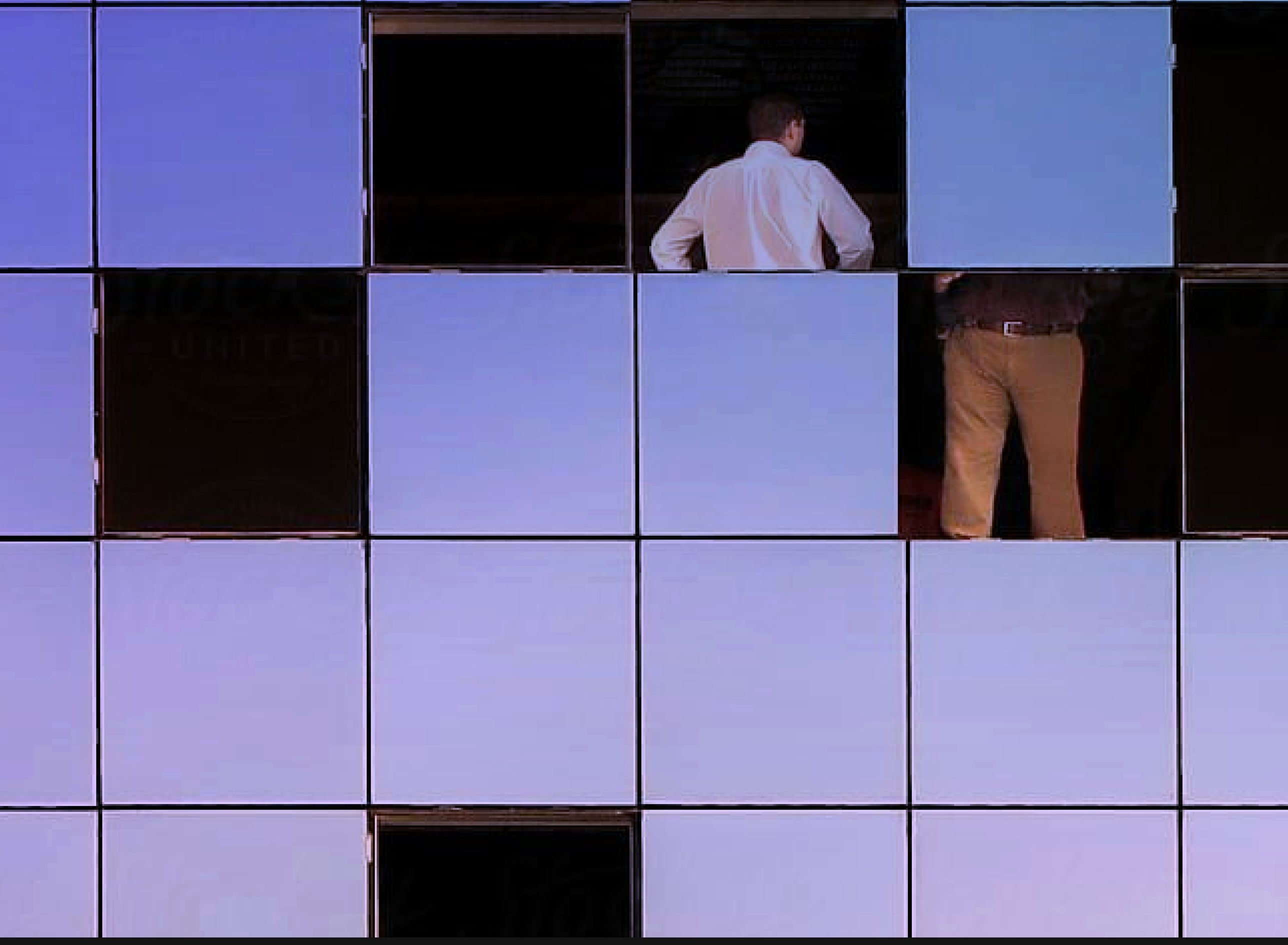
eBook

Five common Jenkins scaling issues and how to fix them



CloudBees

Jenkins® is one of the most widely used development tools on the planet, and it's great at helping small, agile development teams integrate code multiple times a day.

However, developers and coding agents are now pushing far more change through Jenkins than most setups were built to handle, and the strain shows up in several places at once. Maintaining Jenkins pipelines quickly becomes a burden, and visibility starts to slip, so you might lose sight of how many teams are on each instance, how many controllers are in use, or who or what has made changes to code and scripts.

Costs rise as well. In our 2026 State of Code Abundance report, 54% of enterprises said their CI/CD infrastructure spending increased significantly over the past year. In many organizations, the source of all three problems is the Jenkins setup itself. These challenges often surface as an organization reaches the enterprise level and seeks to scale Jenkins company-wide, and a setup that worked for a handful of teams starts to strain under agent-driven change.

Fixing it matters more now, since every new team and every coding agent adds load to the same weak points. As the largest contributor to the Jenkins project, CloudBees has identified five common reasons enterprises struggle to scale Jenkins, along with practical steps to address each

Five signs you've outgrown how you run Jenkins

1. Your Jenkins instance is fragile, so you avoid upgrades and changes.
2. You have lost track of (or control over) which plugins are being used or should be used.
3. You experience multiple outages and restarts that take 20 minutes or longer.
4. Your teams are on their own "islands of Jenkins."
5. You have security concerns, and credentials and permissions are out of control.

This guide explains each of these challenges and how to fix them. Together, those fixes can help you extend your existing Jenkins investment into an enterprise-grade environment that makes software development repeatable, governed, and scalable.

1.

Your Jenkins instance is fragile, so you avoid upgrades and changes.

Jenkins instances often become fragile when a single controller serves too many teams with different needs. Once one team sees results with Jenkins, others usually want in, and the easiest option is to add them to the instance that's already running. Before long, one Jenkins instance is shared by multiple teams with conflicting plugin needs and maintenance schedules.

As developers and coding agents push more commits through that shared controller, each build increases its load, and every upgrade or plugin change puts more teams' pipelines at risk. The longer those upgrades get put off, the bigger and riskier the eventual change becomes.

How to fix it

To achieve the ideal balance between freedom, security, and manageability, we recommend a controller-per-team model with no more than 75–100 jobs each.

This team-based approach boosts the performance and speed of your build jobs and reduces cross-team friction when managing plugin versions and build agent needs. It also limits the blast radius when something goes wrong, so a bad upgrade affects one team instead of the whole organization.

2.

You have lost track of (or control over) which plugins are being used or should be used.



Every team has its own version of Jenkins and its own build processes, including the plugins it prefers. An accurate plugin inventory is one of the best ways to keep Jenkins stable and performing well, but all those different setups make one hard for administrators to maintain. The fewer plugins you have, the lower your upgrade risk and the less testing each upgrade needs.

Plugin sprawl grows with every new team and controller, since each brings its own set of plugins until nobody can say for sure what's installed. More builds also mean that a single outdated or untested plugin fails more often and in more places.

Can you answer the following questions easily? If not, it's time to review your plugin usage.

- Can you list the names of your installed plugins across all teams?
- Do you know how often plugins are being used?
- Which controllers is each plugin installed on?
- What projects are currently using each specific plugin?

How to fix it

Start by building a plugin inventory across every controller that shows which projects use each plugin and how often. Then standardize on a curated set of tested plugins for every team to start from, and remove the ones nobody uses before your next upgrade.

CloudBees CI makes this easier to maintain. The CloudBees Assurance Program gives teams a set of plugins that CloudBees has tested for stability, security, and compatibility. Plugin usage reporting shows which plugins are actually in use across your controllers, so you know which ones are safe to remove.

3.

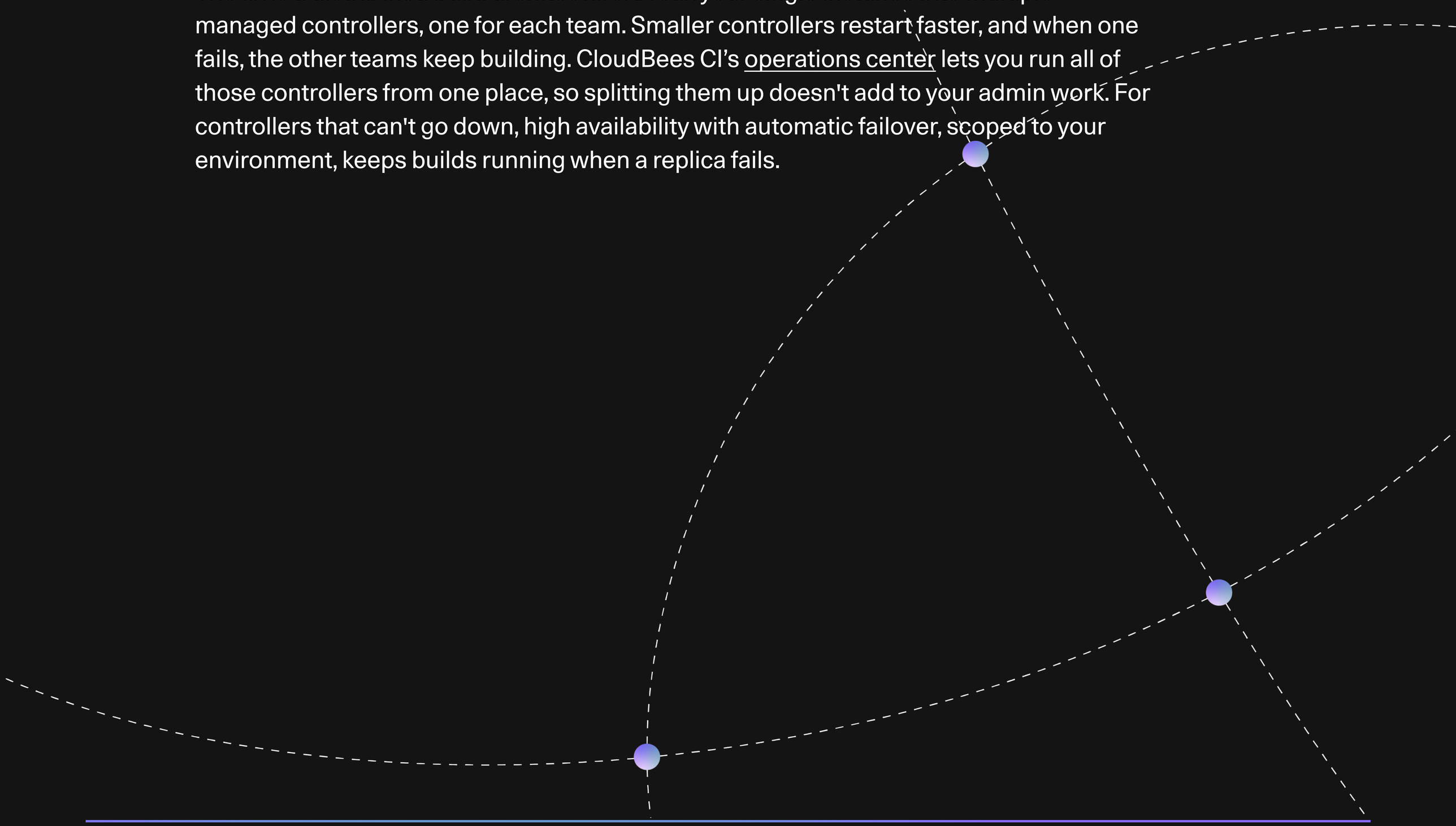
You experience multiple outages and restarts that take 20 minutes or longer.

Frequent outages and long restarts are another sign of a poorly managed Jenkins instance. The more jobs, users, and plugins your instance has, the longer it takes to start up. Poorly written or complicated pipeline code can also slow Jenkins down and limit how far it scales.

As developers and coding agents push more changes, more builds land on the same controller at once, which often pushes it past its limits and causes an outage. Each restart takes longer as the instance grows.

How to fix it

The fix is a distributed build architecture. Break your single instance into multiple managed controllers, one for each team. Smaller controllers restart faster, and when one fails, the other teams keep building. CloudBees CI's operations center lets you run all of those controllers from one place, so splitting them up doesn't add to your admin work. For controllers that can't go down, high availability with automatic failover, scoped to your environment, keeps builds running when a replica fails.



4.

Your teams are on their own "islands of Jenkins."

A controller per team solves the problem of one overloaded instance serving everyone. But with so many separate servers, each team ends up on its own island.

On these islands, engineers maintain different Jenkins instances and configurations. Each team patches, upgrades, and configures its own controller, and a fix one team finds has to be rediscovered by the next.

Governance is uneven: Some teams build security scans into every stage of their pipelines, while others skip them. Administrators have no central view of what each server is doing and no way to enforce the same rules across all of them.

Good practices also stay stuck on one island. When a team builds a better pipeline or configuration, every other team has to recreate it by hand on its own controller, so most never do.

Agentic coding amplifies all of these problems, since every gap between controllers now affects far more changes than before.

How to fix it

Keep the controller-per-team model, but connect the islands. Standardize configuration with configuration as code (CasC) so every controller starts from the same approved setup, and use shared pipeline templates so practices like security scans are built in for every team. Then manage every controller from one place, so you can see what each one is doing and roll out changes to all of them at once.

CloudBees CI's operations center gives you that single view, while CasC bundles and the Pipeline Template Catalog let you apply the same configuration and pipelines across teams.

5.

You have security concerns, and credentials and permissions are out of control.

One of the best things about Jenkins is how easy it is for anyone to set up a server and start integrating code. One of its biggest drawbacks is exactly the same thing. Users are often onboarded without clearly defined roles or team membership, and without those, controls like role-based access control (RBAC) don't do much.

When it's time for an audit, this lack of governance can become a nightmare. With so many Jenkins instances in place, pulling together a record of every action taken, by user and role, along with every change to every job and configuration, becomes its own project. Teams can end up spending more time preparing for audits than delivering software.

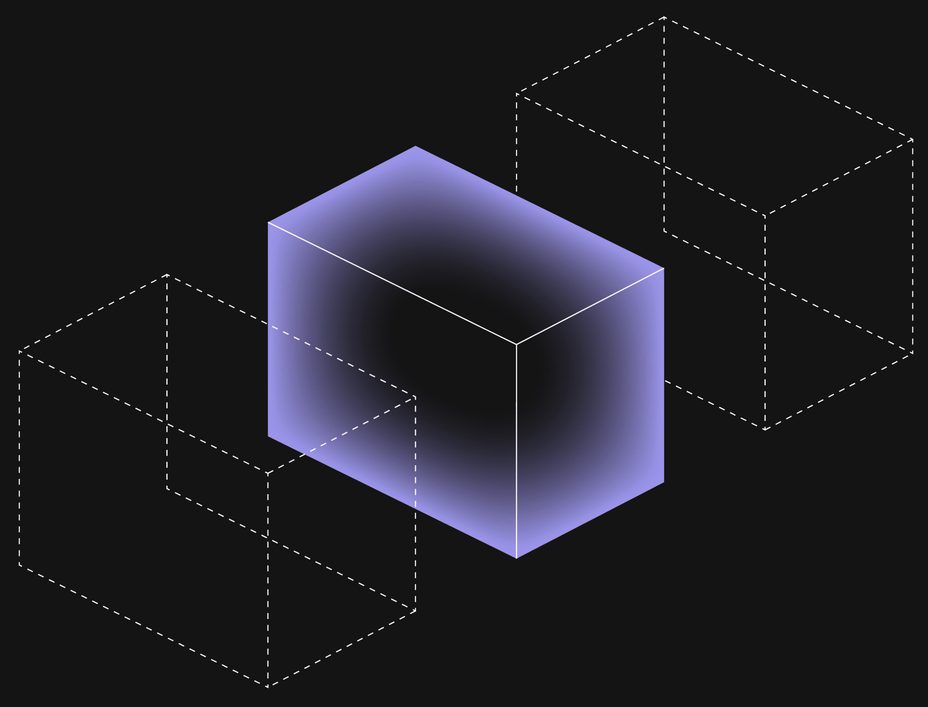
Agentic coding adds to this problem as well. Agents and other automated tools need credentials and permissions, too, and auditors want to know whether a person or an agent made each change. If access was never clearly defined for people, it's even harder to control for agents.

How to fix it

Define roles and team membership when users are onboarded, and give agents and service accounts their own clearly scoped permissions. Then enforce RBAC centrally across every controller so the same rules apply everywhere, and keep an attributable record of pipeline changes: who changed what, when, and under what policy.

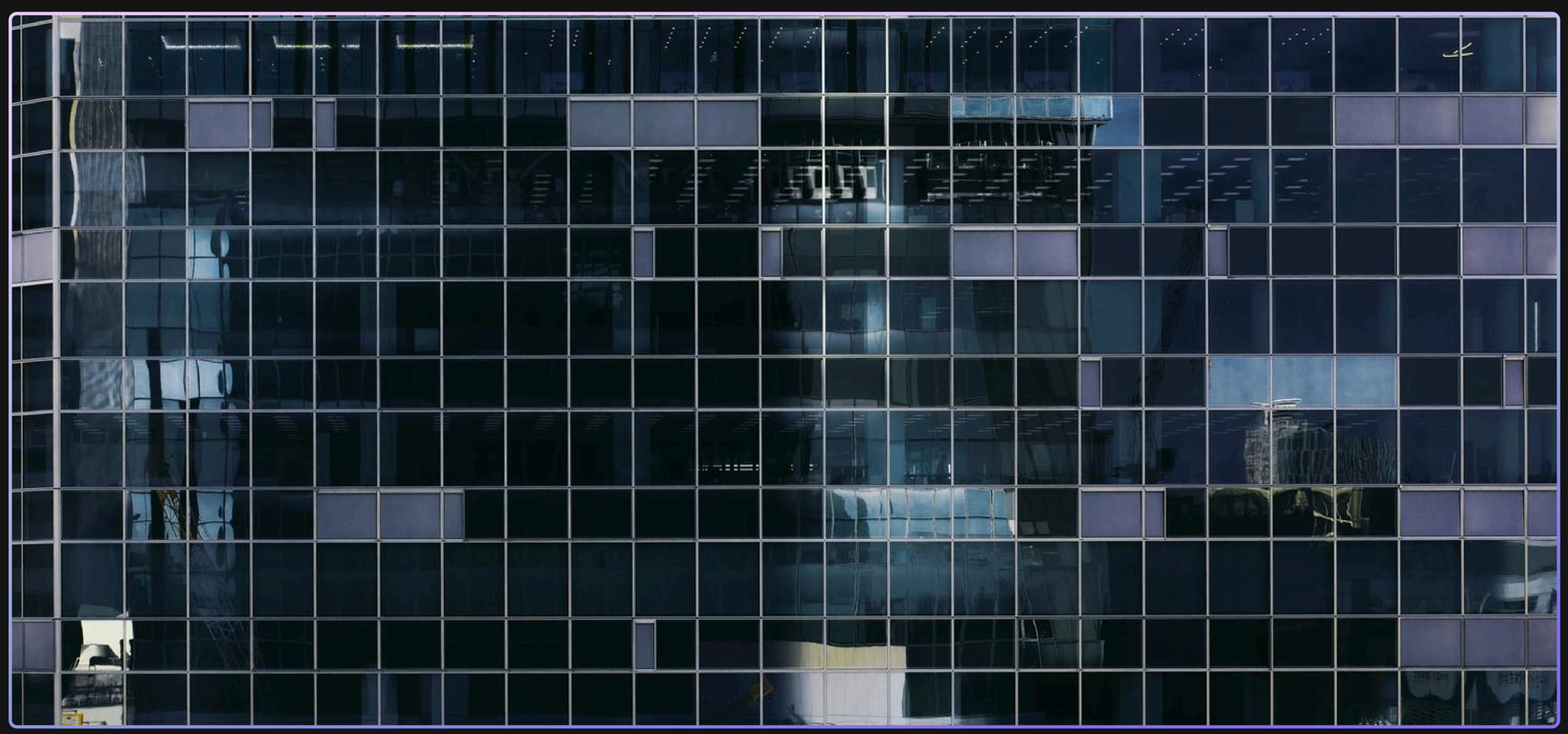
In CloudBees CI, operations center lets you set roles and permissions once and apply them to every controller, so people, service accounts, and coding agents get least-privilege access, and you can see who changed what from one place.

How CloudBees CI helps enterprises scale Jenkins



CloudBees CI is built on Jenkins, so your teams keep the jobs and pipelines they already use. It adds the central management, governance, and expert support that make each of these fixes work across every controller at once. You keep your Jenkins and scale it to the pace coding agents set.

Read about how Autodesk saw a 10x productivity increase with CloudBees CI.



About

CloudBees helps organizations build, run, and govern software factories, giving enterprises the confidence to ship software better, faster, and safer. As enterprises adopt agentic coding, software is created faster than most teams can review, secure, and validate. CloudBees addresses this challenge without asking teams to replace the tools they already use. Across every toolchain a customer runs, CloudBees makes each change, human or AI, visible, auditable, and accountable before it reaches production. Founded in 2010, CloudBees is backed by Goldman Sachs, Morgan Stanley, Bridgepoint Capital, HSBC, Golub Capital, Delta-v Capital, Matrix Partners, and Lightspeed Venture Partners.

Visit CloudBees at www.cloudbees.com.

Jenkins® is a registered trademark of LF Charities Inc.
Read more about Jenkins at: www.cloudbees.com/jenkins/about

© 2026 CloudBees, Inc., CloudBees® and the Infinity logo® are registered trademarks of CloudBees, Inc. in the United States and may be registered in other countries. Other products or brand names may be trademarks or registered trademarks of CloudBees, Inc. or their respective holders.

CloudBees