



1 010000000

1 010000000

Massive

1 001 101

ROI



with Massive Jenkins®

1 010000000

Table of Contents

Introduction	03
Flexibility Has Its Price	04
Imperfect Solutions Have Their Price	09
How Managed Jenkins Can Help	13
CloudBees CI for Maximum ROI	19



Introduction

The power and dominance of Jenkins® for continuous integration (CI) is indisputable—it's used by just about everyone everywhere to continuously build and test software at scale. Equally indisputable? It can be a handful. Extensibility is the foundation of its power, but limitless possibilities can make it unruly. Teams striving for digital and enterprise transformation are stretched to the limit simply trying to realize their goals; there's no room for inefficiency which could result in increased stress, lost productivity, and budgetary waste.

Throw in the potential for regulatory non-compliance and unchecked security vulnerabilities, and Jenkins—if not effectively managed—can create as many problems as it solves. For the many enterprises dependent on Jenkins, there's a vital need to harness what's great about Jenkins without enduring the downsides. Equally vital? Achieving this goal while honoring Jenkins' open source spirit: Solutions shouldn't cost an arm and a leg.



Flexibility Has Its Price

Jenkins dominates enterprise SDLCs thanks to its unmatched flexibility; with 1,800+ plugins extending functionality and tool integration to every corner of development landscapes, there's very little Jenkins can't do. But two systemic issues have the potential to negatively impact development teams and the enterprises they support:

- **Lack of centralization.** Jenkins lacks centralized control. Enterprises that spin up multiple controllers are obligated to govern those controllers individually. There's no hub from which operations, workflows, and processes can be administered across your CI environment, and no single screen from which users can be authenticated and provisioned globally. Orchestrating plugins or projects spanning multiple servers from a single cloud interface? Forget about it.
- **Lack of operational visibility.** Jenkins provides no "bird's-eye view" that allows enterprises to see statuses or measure progress across their operations. In fact, even knowing who owns a project can be difficult. As a result, disconnected teams often work from incomplete info in siloes, and managers struggle to coordinate the big picture across the SDLC.

These shortcomings have sweeping consequences, resulting in four essential "woes" that all unmanaged (i.e., strictly open source) Jenkins controllers must negotiate.





How much administrative overhead are we talking?

For Esri and Office Depot, the admin burden was significant:

- Esri: 30% higher admin burden w/ open source Jenkins
- Office Depot: 50% higher admin burden w/ open source Jenkins

Woe #1: Excessive Admin Overhead

Jenkins' potential for administrative friction can imperil productivity for both engineers and admins. After all, if your admins exhaust all their time chasing down these issues, they'll inevitably rely on your best engineers to pick up the slack.

- **Managing all of your controllers is complex and time consuming.** Jenkins, like any Java application, is not a “set it and forget it” technology. Controllers require continuous administrative oversight, from ensuring proper garbage collection to managing memory utilization. Simply maintaining and updating servers is a job unto itself, and the demands multiply with every server you spin up.
- **Managing, replicating, and provisioning Jenkins instances across teams is difficult.** Different teams have different needs, and launching and maintaining multiple purpose-built servers, provisioning non-standardized plugin sets to each, and supporting different versioning schemes to meet distinct compatibility and security concerns can be exhausting for central admins and shared services teams—particularly if they're attempting any form of governance or compliance.
- **Scaling can stress admins as much as it does infrastructure.** As teams experience success with Jenkins, other teams will want in on the action. How will an indeterminate number of Jenkins controllers (running different toolsets) impact disk utilization, garbage collection, or memory availability? Will the plugins in play impair performance? Will you need more infrastructure to handle the ever-changing load? These are the questions that'll [keep your admins up at night](#).
- **Plugins don't maintain themselves.** Admins spend considerable time updating (or deliberately not updating), negotiating compatibility conflicts, implementing bug workarounds, and resolving security issues as plugins multiply across controllers and teams.
- **Jenkins requires advanced JVM know-how.** Exacerbating all of the above, Jenkins admins need JVM-focused skills that they often lack. This makes it more likely that engineers will be pulled into admin roles in the interest of pooling JVM knowledge to resolve issues.



Woe #2: Inefficiency

When you spend time doing the same tasks over and over again, that's a problem. Inefficiency can afflict team productivity, hinder release schedules, and threaten goals across entire enterprises.

- **Work is unnecessarily duplicated across teams/servers.** Once you've figured out a controller setup that works, there's no way to quickly iterate that across additional servers/teams. This means setting up the controller, adding users, installing plugins, etc., from scratch every time. In fact, workflows of all kinds tend to be repeated again and again.
- **Infrastructure is often wasted.** Without a means for overseeing and balancing resource utilization, infrastructure is often allocated where it isn't needed. This could be from redundant controllers, outdated plugins, poor performance due to lack of JVM maintenance, or wasted idle time.
- **Workflow efficiencies are hard to replicate.** If a team comes up with a clever setup, workflow, or process innovation, there is no simple way to share those enhancements with other teams on different servers. As a result, enterprises lose valuable opportunities to increase productivity across the board.
- **Onboarding can be tedious.** Setting up new teams requires the same repetitive series of tasks every time. This creates significant drag that limits early-stage productivity.
- **Lack of communication can be crippling.** Without visibility across controllers/teams you'll rely entirely on manual communications—an ugly prospect for most enterprises. This can be especially perilous when teams update plugins, tweak their pipelines, change network or firewall settings, or initiate security updates without informing other teams. Things tend to break; it isn't uncommon for teams to submit support tickets along the lines of, "It broke even though we didn't change anything!"

1010 11 001001



“When we first adopted Jenkins, we gave every developer unfettered access, so everyone started using the plugins of their choice. This open access led to version and dependency issues. There was no standard set of plugins, and we lacked a plugin update strategy, which created chaos as we approached our delivery dates.”

Geir Engebakken,
Chief Consultant / DevOps Engineer,
Tietoevry

Woe #3: Inconsistency

At this point, you may notice a theme emerging: decentralized control and lack of visibility generate chaos. With too many parties manipulating too many variables, it's rough getting everyone and everything on the same page. This creates significant roadblocks:

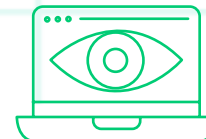
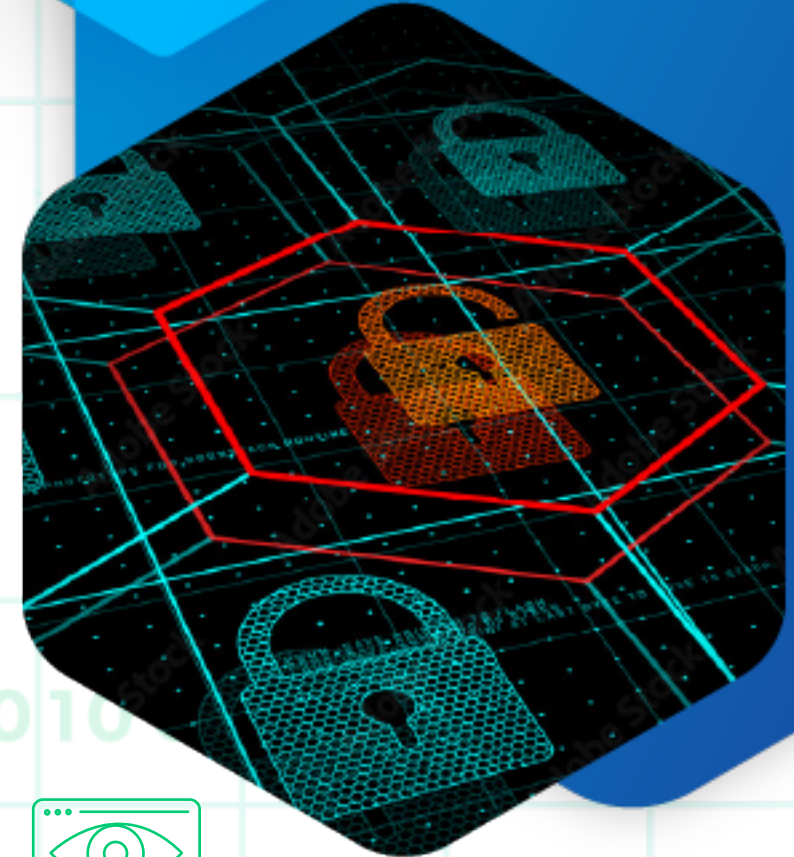
- **Lack of governance inhibits best practices.** When any number of people can spin up, configure, or modify a server, you effectively have no capacity for governance—and little chance of consistently implementing best practices (e.g., regular security scans, prohibitions on unsafe plugins, etc.).
- **Plugins divide servers, workflows, and teams.** Without plugin management, your teams may populate their servers with any number (and quality) of plugins. This opens considerable compatibility and interoperability issues between teams who will inevitably use conflicting plugins and update shared plugins on different schedules.
- **Lack of consistency leads to more bugs.** Pipeline consistency is critical for avoiding bugs; with differently configured controllers, disparate plug-in sets, variable workflows, and largely disconnected teams, unmanaged Jenkins can increase the time your engineers spend hunting bugs.
- **Backups can be sporadic.** Backing up data—and verifying the integrity of those backups—requires discipline. Without meaningful governance, your backup strategy is likely to have holes, and—if that's the case—you're always at risk of losing data.

1010 11 001001

Woe #4: Insecurity

Cybersecurity is an endless challenge. Cloud and business transformations make for highly fluid (and expanding) attack surfaces, and the rapid pace of innovation requires constant vigilance to thwart would-be attackers. So, how does that work for enterprises subject to the woes we've already covered?

- **No one is empowered to enforce security policy.** Without centralized authority or reliable governance, you'll struggle getting teams to adhere to security best practices (or even follow basic policy). This brings risky plugins, risky code, and risky practices to your pipeline.
- **Your security is only as strong as your least safe plugin.** As plugins proliferate across your controllers, any security holes in those plugins become your liabilities. Whether plugin developers plug those holes, and whether your teams reliably update their plugins to take advantage of security fixes, plugins can be security wildcards if not maintained.
- **Vulnerability remediation can be slow to nonexistent.** With too many cooks in the kitchen and a generalized lack of visibility and coordination, implementing fixes for any security threats you do discover will be cumbersome.
- **Chaos plus weak security makes compliance difficult.** Compliance is all about demonstrating that you are in control, have reliable safeguards, and can meet security standards. For all the reasons listed above, this is tough without a managed Jenkins implementation.



Imperfect Solutions Have Their Price

As the woes above ripple through your enterprise, you're likely to experience a series of downstream effects, each with implications for how your pipeline works, how quickly you can release code, and how much the whole operation costs.

Ideally, all such concerns would be tightly controlled to advance your business goals, but things [seldomly play out that way as enterprises scale.](#)



Islands of Jenkins, Jenkinsteins, or Both?

The first, and most profound, consequence of unmanaged Jenkins' shortcomings is that they will inevitably dictate how your Jenkins environment is structured. Jenkins controllers tend to fall into three organizational traps, each intended to prioritize certain concerns at the expense of others. Bottom line? These approaches represent band-aid solutions that don't solve the underlying systemic flaws they are intended to circumvent.

Islands of Jenkins—The “Too Many Controllers” Problem

Because unmanaged Jenkins can make collaboration and standardization difficult, it's natural for teams to want to do their own things. If permitted to, they'll spin up their own controllers, customize their toolchains, and maintain their corner of your SDLC as they see fit. This works well enough for a handful of teams, but enterprises with armies of developers will only maximize their Jenkins woes. This “wild west” scenario produces the islands of Jenkins phenomenon—countless disconnected servers/teams, often working at odds, amplify communication, governance, compliance, and security troubles.

Jenkinsteins—The “Giant Monolithic Controller” Problem

Most of the difficulties associated with unmanaged Jenkins flow from trying to manage too many controllers. Many enterprises attempt to solve this by putting everything on one controller. While this does alleviate some maintenance, governance, and compliance concerns, it creates new problems in their place:

- A single server cannot be all things to all teams. Some teams will go without preferred customizations and even essential plugins (if they introduce compatibility conflicts).
- Running all your projects from a single server is likely to overload the server, slowing build and test times enterprise-wide.
- A single server becomes a single source of failure; a server outage breaks your entire organization's productivity.

Jenkinsteins + Islands—The “Worst of Both Worlds” Problem

Many enterprises start with a monolithic approach before eventually buckling under the need to give teams free rein. This combo of monolithic plus rogue servers makes for a particularly chaotic Jenkins environment that combines the downsides of both scenarios while eliminating most of their upsides.

1 0 1 0 1 1 0



Who's Handling Support?

As we explore the challenges presented by unmanaged Jenkins, it's important to remember that community support and your own team members are your only avenues for help if you run into trouble. Open source communities are inspiring examples of people helping people, but they aren't exactly enterprise grade. This can lead to several new sources of frustration:



Enterprises targeting rapid growth/transformation require 24/7, authoritative support to maintain pace. A lack of dedicated, reliable support services will create support bottlenecks that invariably inhibit enterprise growth.



In-house support can become an out-of-control expense. As your teams multiply, the amount of time team members spend on self-help support will also multiply. How much this costs in lost productivity, budget, and growth is anyone's guess.



Downtime can cripple enterprises, generating significant losses. This becomes a real risk if adequate support staff aren't on standby when catastrophe strikes.

1 0 1 0 0





What Is this Costing Us?

Why do all these issues matter? Because they ultimately boil down to a price tag. Worse, that price tag is often invisible. You usually know how Jenkins is helping your bottom line—but you can't always see how it's hurting it. Items needlessly digging into your budgets may include:



Productivity losses due to administrative overhead, troubleshooting, poorly maintained servers, security breaches, etc.



Scheduling delays because your [engineers may spend 15+ hours a week on admin and support instead of code.](#)



Work hours wasted on repetitive tasks and hopeless governance and compliance pursuits.



Missed business opportunities because you're focused on pipeline challenges instead of innovation.



Staff bloat from trying to brute-force your way past the issues above.



Scalability bottlenecks—How can you scale CPU, RAM, and disk space as needed when chaos defines your SDLC? You'll either pay for unused/idle resources or have your growth inhibited by a lack of resources when you need them.



Higher infrastructure bills—How can you recapture costs from idle servers if you don't know when they're active?



Downtime from any number of sources—Broken Jenkinsteins, unidentified bugs, compatibility conflicts, cyberattacks resulting from risky code, etc.

How Managed Jenkins Can Help

We've highlighted Jenkins' shortcomings strictly out of love. Jenkins remains the most valuable CI tool available; there's a reason 80% of enterprise companies use it. You simply need to remedy its shortcomings to unlock its true power and potential. But what does your ideal toolkit look like—how do you amplify the good while ejecting the bad? Even better, how do you do that while saving money?



For an inside look at the sweeping improvements centralization can bring to DevOps pipelines, [read Gurushyam Mony's \(Director, DevOps and Quality Engineering, Markel\) account](#) of how moving Markel's CI pipeline to a cloud management platform saved his dev teams thousands of hours a month.

0100



Centralized Management to the Rescue

Short answer? You need a management platform that eliminates friction and reduces hidden costs in your Jenkins infrastructure. When you establish centralized authority from which all your controllers can be managed, you also unlock exciting visibility, optimization, and governance possibilities because you're now pooling information and functionality to a single portal. What would that look like? Ideally, a centralized control plane capable of the following game-changing feats:

Single-Screen Command of the Entire Jenkins Infrastructure

- Visibility across all controllers, projects, and teams
- Infrastructure health monitoring and alerting
- Instant controller provisioning (supporting individual teams or team collaboration)
- Creation of shared agents
- Centrally managed authentication and authorization (SSO, SAML, RBAC, LDAP)
- Sharing of events or messages across controllers and projects
- Support for connecting, reusing, and managing resources as a cluster of connected assets

Enterprise Configuration as Code (CasC)

- Easily manage, configure, replicate, and update from a single source
- Everything as code: the control plane itself, controllers, agents, projects, plugins, and pipeline templates

Plugin Management

- Centralized plugin provisioning, configuration, monitoring, and updating—with recommended actions to assist with maintenance
- Plugins added to CasC bundles for single-source updates
- Plugin updates via cluster operation

Does this all sound too good to be true? Implementing the features above would go a long way toward wrangling the chaos Jenkins can propagate across your SDLC, but we're just starting to explore the possibilities with your managed-Jenkins feature set.



“Standardizing your applications is where you really start getting the benefits of reusing automation, code structures, and ways of working. This is how an organization really starts to gain speed in the market as it’s trying to compete with other companies.”

Principal Software Engineer,
SwedBank



Standardization, Automation, and Optimization

After you’ve reined in the chaos, it’s time to get serious about improving efficiencies across the board. That means consistency and automation. To amplify and capitalize on the management improvements you’ve brought to your Jenkins landscape, you’ll want your toolkit to include the following productivity-focused features:

Rapid Onboarding

- All dev teams start with a trusted, verified version of Jenkins and a curated set of plugins tested for stability and security.
- Repeatable components are provisioned utilizing custom CasC bundles, enabling teams to start with preconfigured and tested environments managed from a single source.

Pipeline Template Catalog to Enforce Consistency and Stability Operations-Wide

- You have a collection of pipelines based on best practices so teams never have to start from scratch; changes to templates are pushed to all team pipelines from a single point.
- Up-to-date template code is fetched every time a job is run, so jobs always reflect the most current standard. Optionally, developers could be restricted to certain templates to enforce governance goals.
- Governance developers could avoid starting from scratch by modifying existing templates to suit new agendas.

Automated Backup and Recovery—Eliminate human error from your backup strategy.

Cross-Team Collaboration

- Manage events across multiple controllers with a powerful publisher/subscriber model; teams subscribe to pipeline events and act on them regardless of location.
- Pipelines work in sync—Different teams can work collaboratively across pipelines using synchronized, automation execution.
- Teams can be isolated by controllers, but still work together to automate your pipelines.
- Every team should be able to safely use different plugins and technologies without the risk of impacting other teams.

Security and Compliance Made Easy

Security and compliance often represent the biggest challenges for enterprises relying on Jenkins for CI; after all, it's difficult to achieve either if you aren't in total control of your development environment, and few developers are. Fortunately, with centralization and standardization added to the mix, security and compliance become relatively easy. With every team using a secure build and approved pipeline configurations, you've already got guardrails supporting your SDLC across the board. Add to that some compelling features made possible by centralized management, and you'll be running a tight ship in no time:

Centrally Managed Security Policy

- Controllers are authorized to run only tested, validated, secure versions of Jenkins. If issues arise with a build, it's easier to replicate, fix, and update that build globally.
- Manage authentication and authorization features from a single location:
 - » Use single sign-on (SSO) across all controllers.
 - » Define security credentials using the role-based access control (RBAC) model and group access at the control plane, folder, or individual controller level.
 - » Enforce access to controllers, agents, or items using folder hierarchies.
 - » Combine RBAC with folder restrictions for added security; decide who can create, modify, delete, and use credentials at each place in the folder hierarchy.
- Scope security to a per-team basis.
- Templated workflows build security into your pipelines.

Compliance Simplified—Simply build compliance by using CasC bundles and pipeline templates.

Minimized Risk From Plugins—Limit teams to a curated set of plugins that are tested for stability and security and confirmed to meet compliance goals.



What used to take developers a few days now only takes 10 minutes to complete. This, of course, has an impact on our internal resources, as developers are now more impactful. This newly cultivated productivity saves the company money and helps everyone to move faster.

Piotr Musial,
DevOps, Manager IT



With a tenfold increase to productivity, [Autodesk](#) offers a great example of the gains enterprises can see when they incorporate these efficiencies into their Jenkins pipelines.

1 0 1 0 1 1 0

Scalability to Match Your Growth

Automating at scale is the Jenkins grail quest. Unmanaged Jenkins is great at the automation part—not so much at the at scale part. Fortunately, our ideal toolkit already includes the building blocks for straightforward scalability; with a few additions to the toolkit, scaling will almost take care of itself:



Deliver your Jenkins at scale by automating provisioning of new teams/controllers, propagating best practices (pipeline templates, CasC), and managing resources for right-sized infrastructure.



Teams are empowered to quickly spin up new controllers and workspaces on their own while ensuring that the SDLC remains consistent, governed, compliant, and secure.



If you are using Kubernetes, **optimize infrastructure resources when scaling** by leveraging Kubernetes provisioning; controllers can be spun up as needed, then hibernated when not in use.

1 0 1 0 1 1 0 0 1 0 0 1

1010 11 001001



Enterprise-Grade Support

Finally, to truly make Jenkins enterprise ready, we have to address the elephant in the room: lack of support with Jenkins. Community assistance is one of the best things on Earth, but when disaster strikes you don't have time to dig around forums to find a solution. To bake resiliency into your game plan you'll need rock-solid, professional support so you can concentrate on building apps rather than chasing down fixes.



How much impact can managed Jenkins have? Enough to transform your entire approach to development. For an in-depth look at how sweeping the changes can be, [read how it helped SailPoint Engineering](#), "keep up with the pace of the company, adding controllers as new products are created or new companies are acquired."



1010 11 001001

CloudBees® CI for Maximum ROI

The next question is obvious enough: where can you get all this? Simple answer: the CloudBees software delivery platform. Part of that platform is CloudBees CI. CloudBees CI provides all of the above and more. As the largest contributor to Jenkins code, CloudBees engineers are the definitive Jenkins experts. The CloudBees team loves Jenkins. But they're well aware of its potential pitfalls, so they're committed to maximizing Jenkins' potential



Productivity gains through CloudBees CI are often dramatic, as [Accenture UK](#) discovered:



Home

Dashboard

Admin

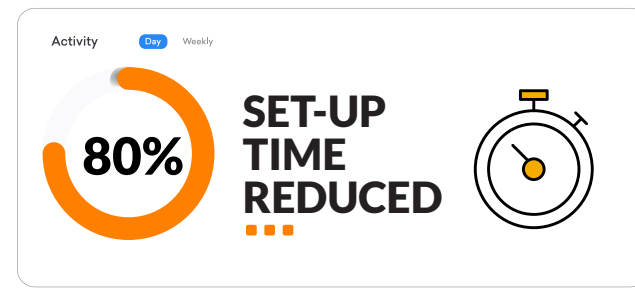
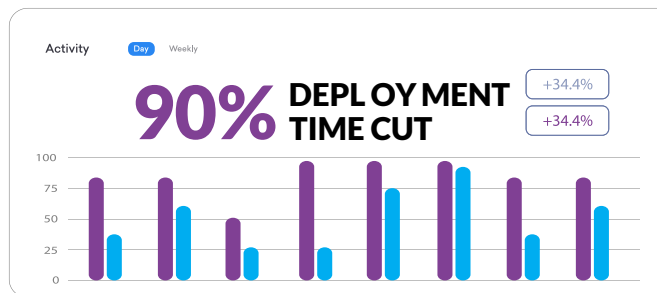
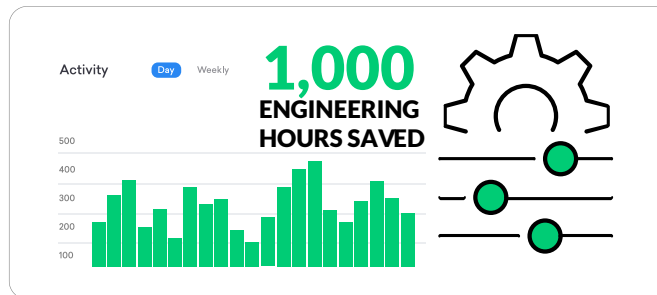
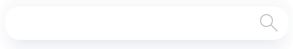
Messages

Settings

Logout








Dashboard **accenture**

Productivity Gains through CloudBees CI



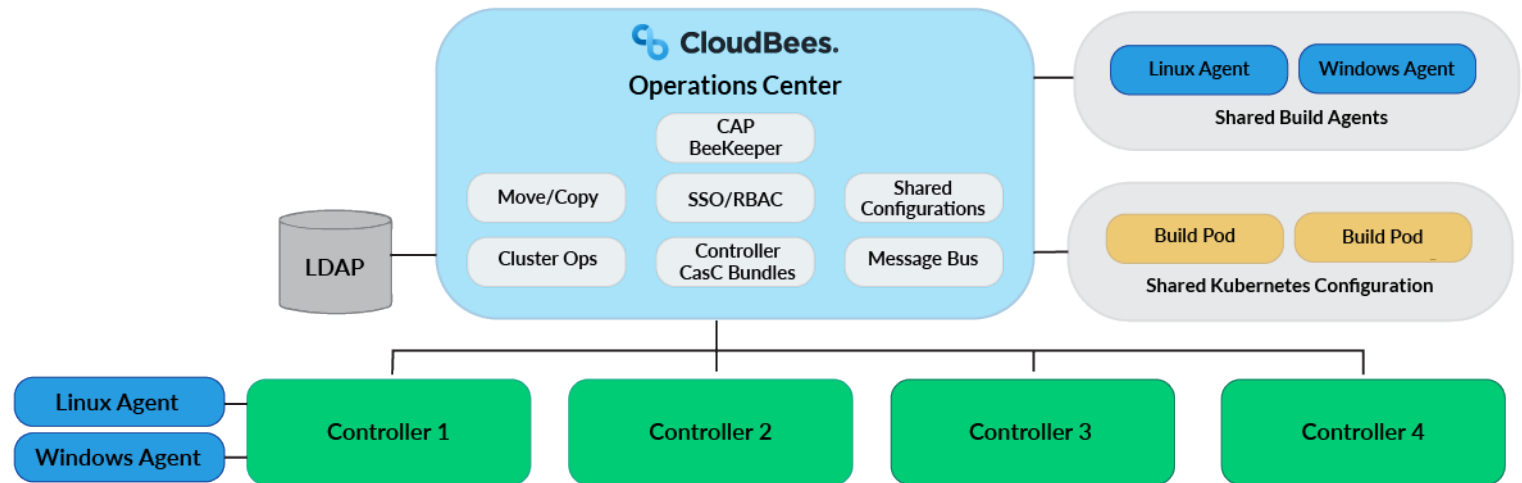
CloudBees CI Operations Center

The diagram below shows how the CloudBees CI Operations Center (the centralized control plane we talked about earlier) restructures your Jenkins environment to facilitate the toolkit we've discussed, whether you deploy on premises or in the cloud.

- 
-  Home
-  **Dashboard**
-  Admin
-  Messages
-  Settings
-  Logout

Dashboard

CloudBees CI: Architecture and Components





“CloudBees CI has given me 30 percent more time in my day.... We’re using the time previously spent on day-to-day manual tasks on higher value projects.”

Vince Ingram,
Senior Systems Admin, Esri



CloudBees CI also offers some important differentiators you won't find with other managed Jenkins options:

Built by the Leading Contributor to Jenkins—Because we contribute a lot of code to the open source project, we are uniquely situated to fix issues. Additionally, CloudBees proactively addresses vulnerabilities in Jenkins to ensure they never reach your installation.

CloudBees Assurance Program (CAP) Plus Beekeeper

- The CloudBees Assurance Program (CAP) starts with a rigorous vetting process geared toward curating a set of enterprise-grade plugins that are safe for teams to use. Not only are these plugins independently stable, but they are tested in aggregate to ensure compatibility with each other and your CloudBees CI controller.
- Once a CloudBees CI controller is enrolled in CAP, the Beekeeper Upgrade Assistant monitors installed plugin versions and reports on instance configurations. With analytics into how every plugin in your controller is being used, you gain valuable insights into areas of waste and opportunities to reduce complexity in your pipeline.
- With the CloudBees Assurance Program, your engineering teams can eliminate hours of technical debt at the click of a button while gaining confidence and insights into the plugins that keep their CI solution running smoothly.

Jenkins Health Advisor—Available to all Jenkins users, the health advisor provides health-status monitoring across your Jenkins environment, delivering notifications before issues (security vulnerabilities, performance bottlenecks, plugin defects, etc.) can affect your users.

Elasticity at the Cluster Level (Modern Cloud Platforms ONLY)

- Instantly provision controllers.
- Dynamically share build agents across controllers.
- Hibernate idle resources.

Pipeline Policy and Best Practices Enforcement — Create and maintain build pipelines with best practices built in. Our Pipeline Templates Catalog helps ensure that pipeline builds conform to organizational standards, while Custom Marker Files allow for instant build-template selection based on software configuration management (SCM) identifiers.

Developer Productivity Enhancements—Receive granular, actionable build data directly in GitHub and Slack.

Customer Success Managers—Customer Success Managers help optimize your CloudBees CI experience with tips, tricks, and updates that ensure you stay on the path to growth.

Professional Services—Whether you're getting started with CloudBees, building DevOps mastery, or moving from old to new, our professional services team will get you there—quickly.

Superior Support

- 24/7 on-call support from the largest group of Jenkins certified engineers
- Assisted updates. Keep CloudBees CI current, stable, and compliant by proactively planning your upgrades with our support team.
- [CloudBees TV](#) contains many how to's for both Jenkins and CloudBees CI
- [CloudBees Docs](#) online documentation provides information and forums to enrich self-help



“CloudBees CI has become the heartbeat of our development team and the central point that everyone uses to see the status of the code base and what is happening with it right now. Having that transparency and visibility into the quality of our code is incredibly valuable.”

Pete Hayes,

Senior Director Developer Productivity Engineering, Pega





Office Depot estimates savings of \$7k+ per developer per year from moving to CloudBees.



1 0 1 0 1 1 0

ROI by the Numbers

CloudBees CI's centralized management and extensive feature set translate to fewer hours worked, less attack surface exposure, rarer performance bottlenecks, faster time to market, and reduced organizational chaos. Additionally, CloudBees offers features that extend beyond productivity, attacking infrastructure costs directly:

For CloudBees CI on both traditional and cloud platforms:

- Leverage CasC for controllers to manage parallel controllers with zero marginal management cost: manage inbound events, cluster administrative operations, and share cloud agent configurations across clusters of controllers.








For CloudBees CI on cloud platforms:

- Lower cloud infrastructure bills by optimizing resource utilization to recapture costs from idle controllers and static agents—automatically set your controllers to hibernate when they're idle and automatically destroy agents after work is completed.
- Leverage Kubernetes provisioning to dynamically spin up or hibernate controllers.

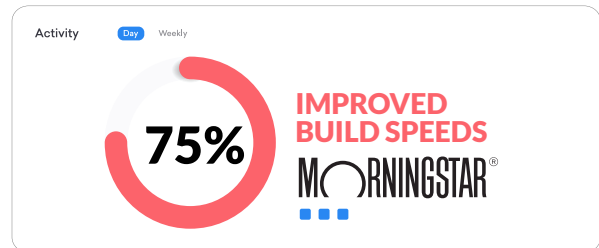
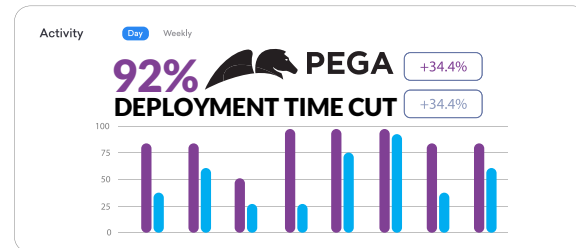
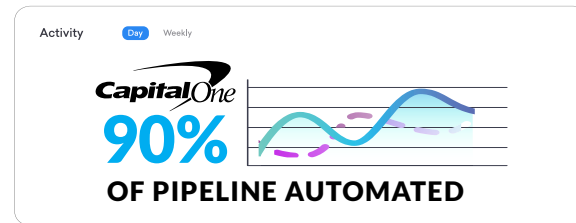
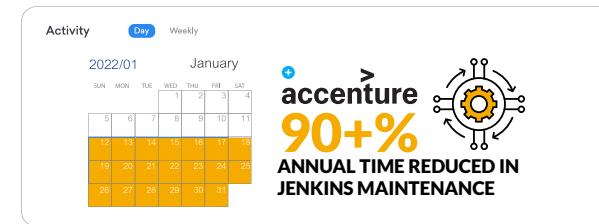
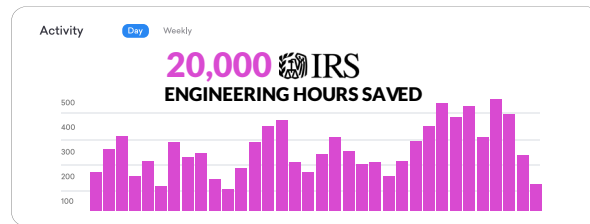
0 1 0 1 1 0 0 1 0 0 1

What Kind of ROI Can You Expect From All This?

The following charts show results reported by CloudBees customers.








- 
-  Home
-  **Dashboard**
-  Admin
-  Messages
-  Settings
-  Logout

Dashboard Productivity Gains through CloudBees CI



CloudBees Customer Base Productivity

The following charts show results from the CloudBees customer base study of productivity gains over time.

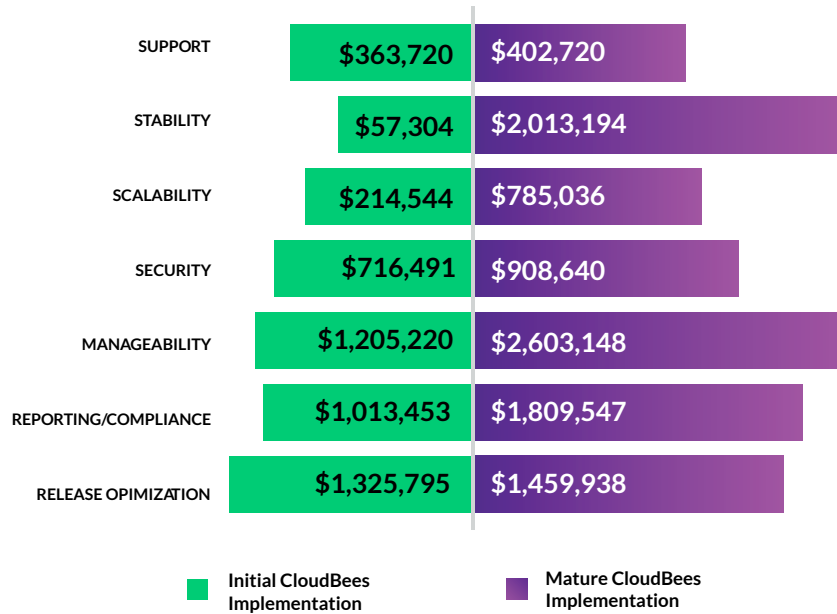
- 
-  Home
-  **Dashboard**
-  Admin
-  Messages
-  Settings
-  Logout

Dashboard

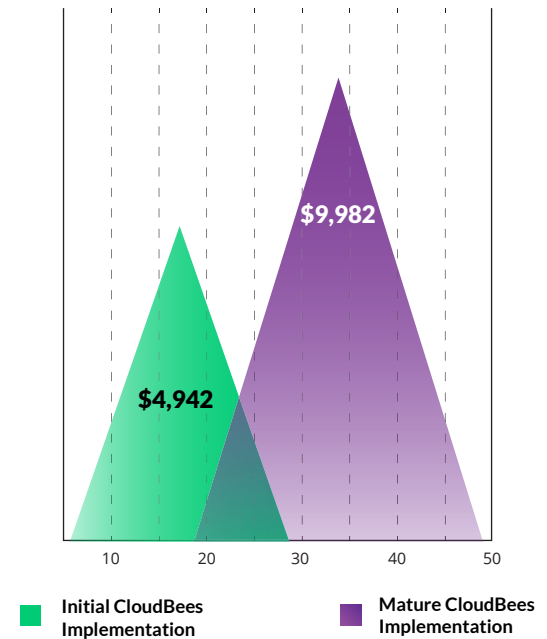
Productivity Study

Go to search

Estimated Annual Recurring Gains by Category - Per 1,000 Users



Estimated Total Annual Recurring Gains - Per User



1010 11 001001



How We Can Help

If you're ready to maximize the potential of your Jenkins implementation, contact us for a [personalized ROI assessment](#). We look forward to speaking with you.



"With CloudBees, we made a conscious decision to bring in a platform that was enterprise-ready, geared toward both developers and operations teams alike. That platform is making it possible for us to simplify the landscape, so our teams can bring innovative capabilities to market faster."

Nick Sbordone,

Director of IT and Digital Platform Engineering, Office Depot



1010 11 001001



About CloudBees

CloudBees provides the leading software delivery platform for enterprises, enabling them to continuously innovate in a world powered by the digital experience. CloudBees enables organizations with highly-complex environments to deliver scalable, compliant, governed, and secure software from the code a developer writes to the people who use it.

CloudBees is helping thousands of companies harness the power of continuous everything and gets them on the fastest path from great idea, to great software, to amazing customer experiences, to being a business that changes lives.

Visit CloudBees at www.cloudbees.com.



CloudBees®

CloudBees, Inc. • 4 North Second Street Suite 1270 San José, CA 95113 United States • www.cloudbees.com • info@cloudbees.com

© 2022 CloudBees, Inc., CloudBees and the Infinity logo are registered trademarks of CloudBees, Inc. in the United States and maybe registered in other countries. Other products or brand names may be trademarks or registered trademarks of CloudBees, Inc. or their respective holders.
Jenkins® is a registered trademark of LF Charities Inc.