eBook

The Engine Inside Successful Software Factories

CloudBees

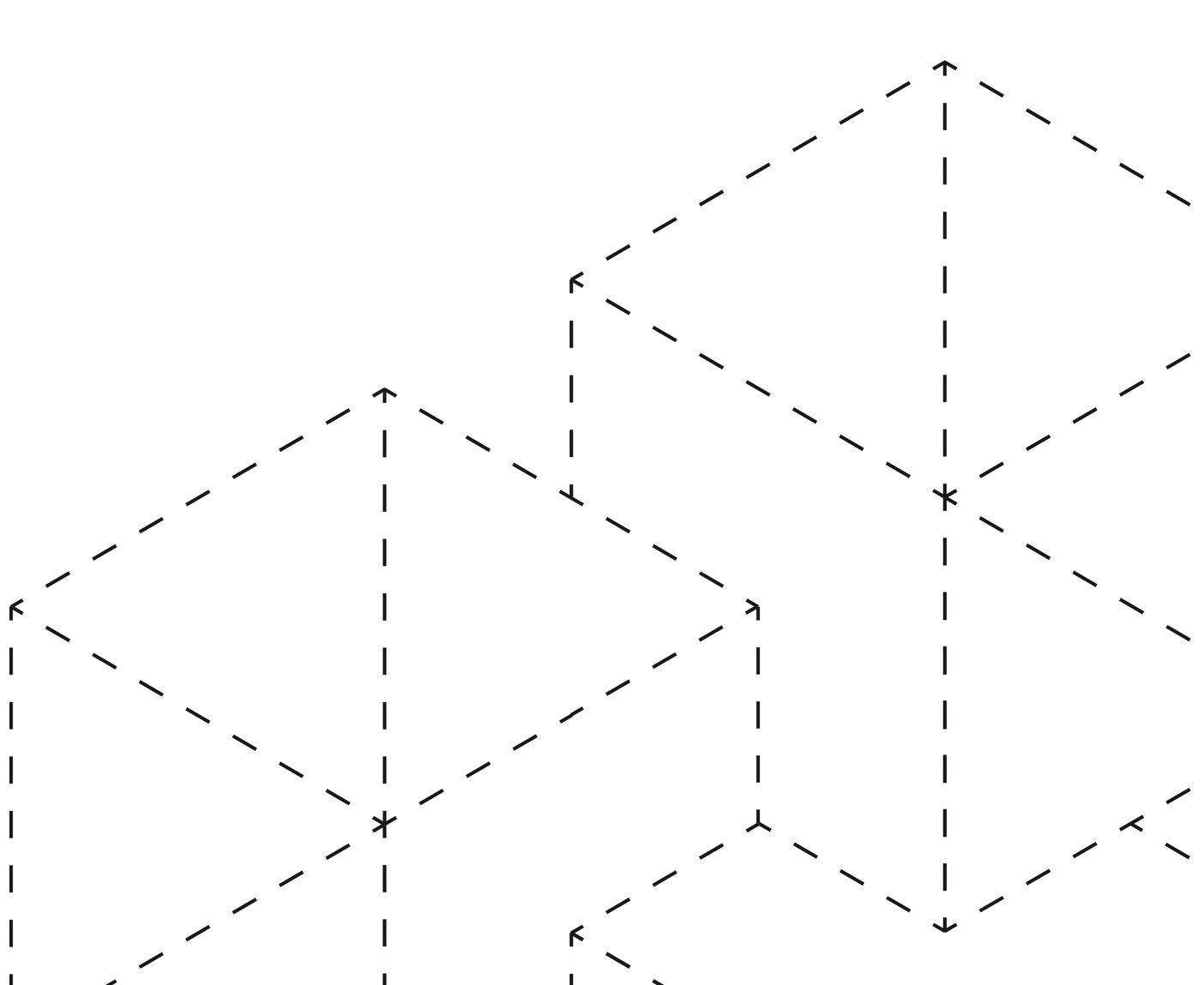
Table of Contents

What is a software factory and why does the government care?

A software factory is an organized collection of software assets, tools and processes that expedite the production and delivery of software solutions. This “factory” approach is modeled directly off of traditional manufacturing techniques where a collection of tools alongside a well designed process eliminates waste throughout the product lifecycle. In the software world, this means eliminating redundant activities, minimizing unnecessary manual tasks and maximizing the value derived from developers. The number one requirement to accomplish these goals is AUTOMATION! In a traditional factory, the assembly line connects the processes, tools and people required to produce high-quality products. Similarly, a good software factory requires automation to connect the processes, tools and people into high-functioning pipelines to deliver software that is always deployment worthy. A common term for this approach is DevSecOps.

By working with a large number of government and private organizations across hundreds of implementations, we see several best practices emerge regarding software delivery and achieving increased quality, security and speed of DevSecOps at scale. Enduring success for these organizations happens when they fold their program management and governance goals into the DevSecOps processes around their toolchains. The programs start to benefit from greater management insight into performance, and they have an easier time scaling their automation initiatives, which drives further efficiency.

Furthermore, they gain the ability to automatically gather data from across their DevSecOps processes and delivery pipelines. This data clearly shows how they are deploying quality software with speed. Ideally, the data is as close to real-time as possible and collected continuously. This “evidence” is vital for things like assessment and authorization (A&A), in accordance with the NIST RMF, to know that your factory is producing “worthy” software – or to pick up on problems. However, it’s not enough to merely tell you there’s a problem; you need to trust that the pipelines will automatically fail any build or deployment that isn’t up to snuff. That is one of the major benefits of automation.



Five Key Strategies Driven by a Software Factory

1

GitOps - Configuration as Code (CasC)

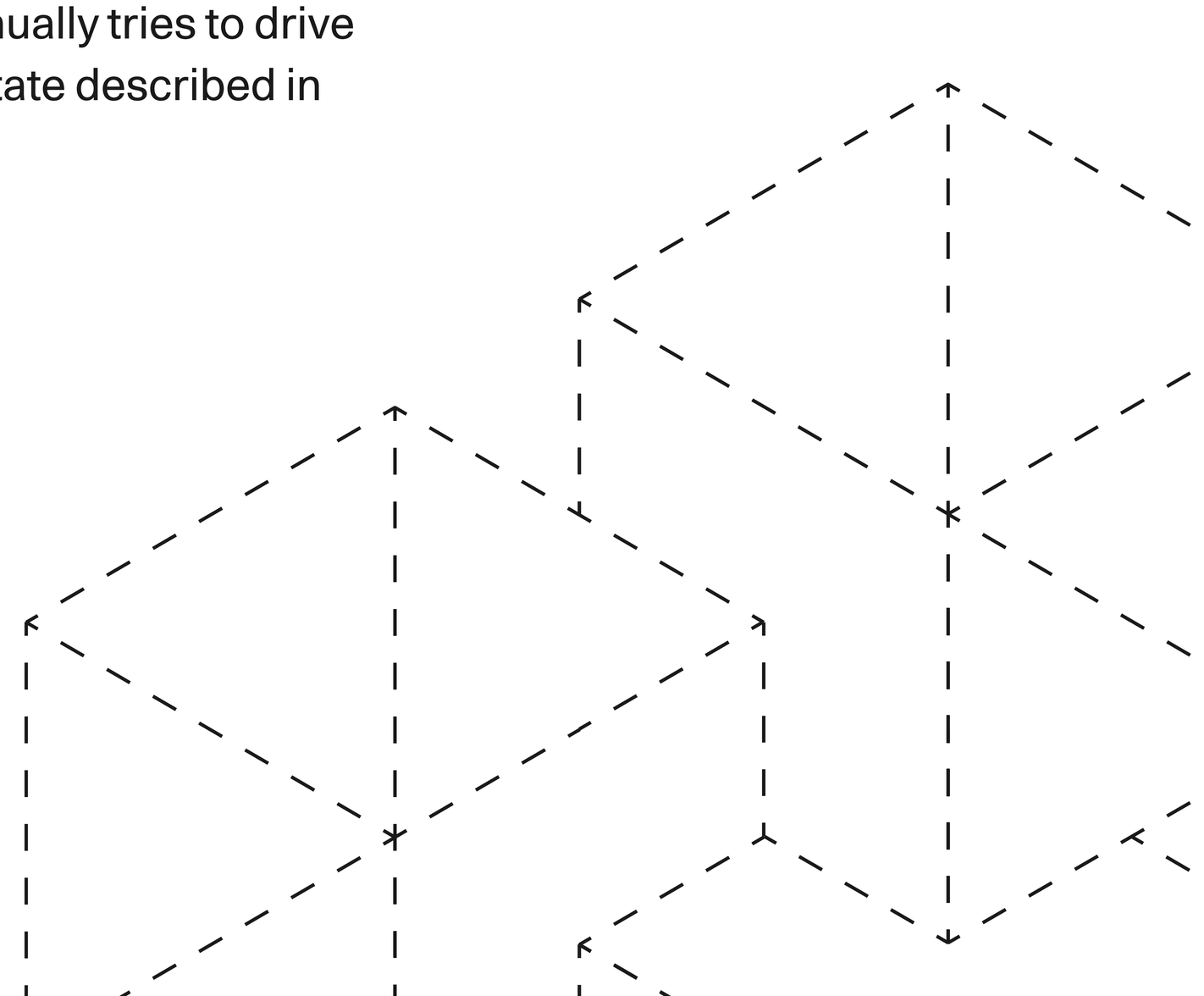
GitOps upholds the principle that Git is the one and only source of truth. GitOps requires the desired state of the system to be stored in version control such that anyone can view the entire audit trail of changes. All changes to the desired state are fully traceable commits associated with committer information, commit IDs and time stamps. This means that both the application and the infrastructure are now versioned artifacts and can be audited using the gold standards of software development and delivery. GitOps is based on a Git-based source code management system and hence GitHub, GitLab and Bitbucket are natural choices for a code repository.

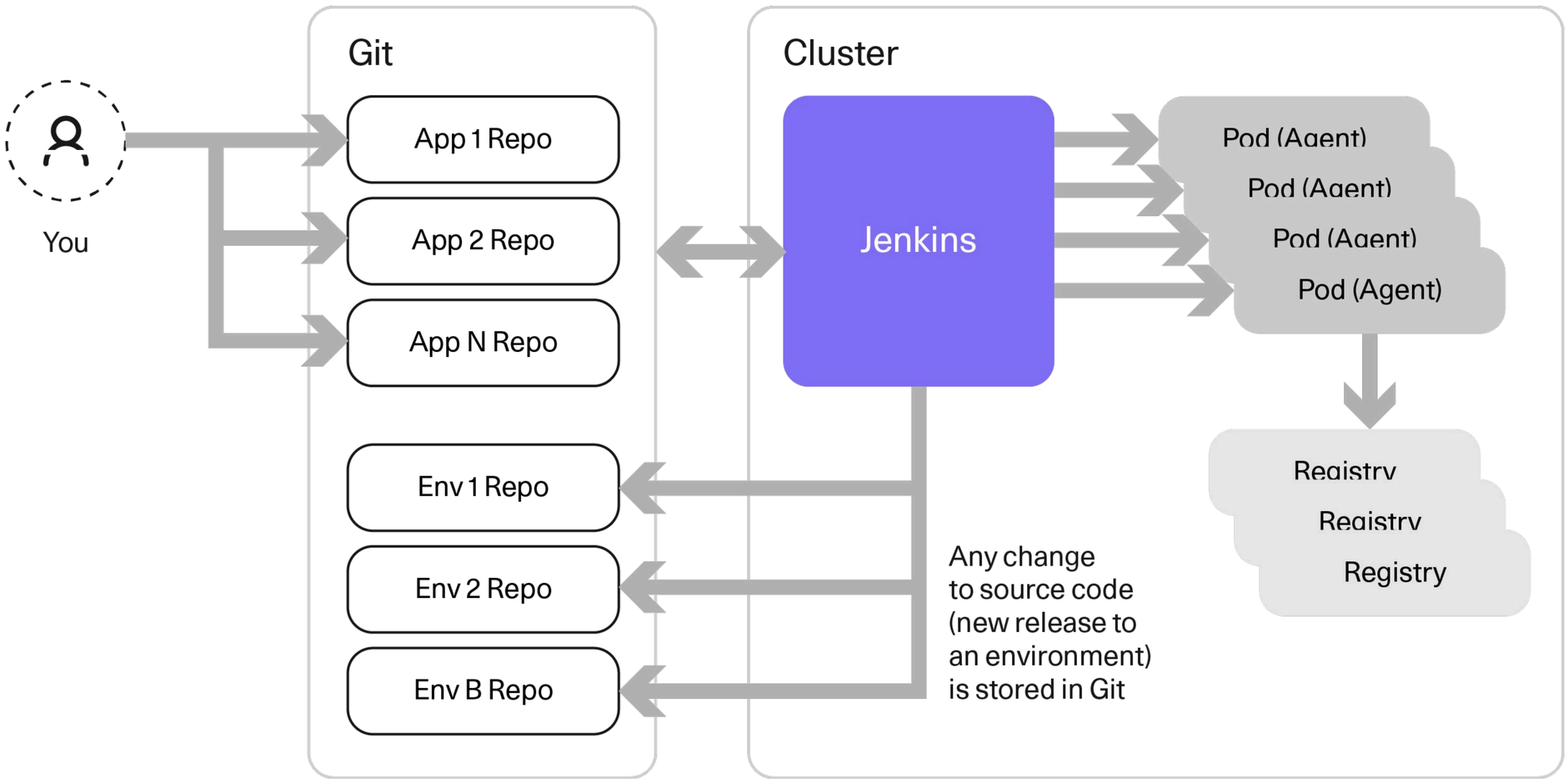
Declarative specification for each environment

GitOps requires us to describe the desired state of the whole system using a declarative specification for each environment. This becomes the system of record. You can describe your environments – such as test, staging and production – in a code repo, along with the application version that resides in that environment.

The relevance for software factories is simple: by having a declarative definition of the whole system, you mitigate the complexity of tasks like audits and rollbacks. Along with those tasks it allows the government to approach cloud migrations, multi cloud and hybrid cloud much like a smaller, leaner organization would. The declarative definitions mean that infrastructure can be modular and redefined across any environment or cloud. This additional flexibility, freedom and governance empowers a software factory to serve a much larger group of programs.

For example, we can describe everything in the Kubernetes model as a declaration. The Kubernetes API server accepts a declaration as an input, and then continually tries to drive (or converge) the cluster to the desired state described in the declaration.





GitOps beyond the app/environment level

System management involves much more than just applications and environments. It also involves the open source and COTS tools that intertwine to create the critical toolchain organizations rely on. It's important that these components also fit within the GitOps model so that they don't inadvertently become a drag on organizations and hinder their digital transformations.

The CI/CD toolchain incorporates an important set of tools – throughout the software supply chain – that make up the technical backbone of a software factory. Oftentimes this consists of myriad technologies performing a wide variety of functions, but at its core the CI/CD engine orchestrates the workflows from code commit through testing, deployment and release.

The situation can become complex when the various teams supported by the software factory have diverse tooling requirements and workflows. How can the software factory satisfy those needs most effectively? With CasC and GitOps.

How can CloudBees help?

CloudBees reinforces the significance of GitOps by capturing all configuration as code. The foundation of CloudBees technology is Jenkins, the industry leading open source CI engine. As a best practice, Jenkins jobs (pipelines) are declared as code. However, the actual configuration of open source Jenkins instances (controllers) does not follow this same pattern.

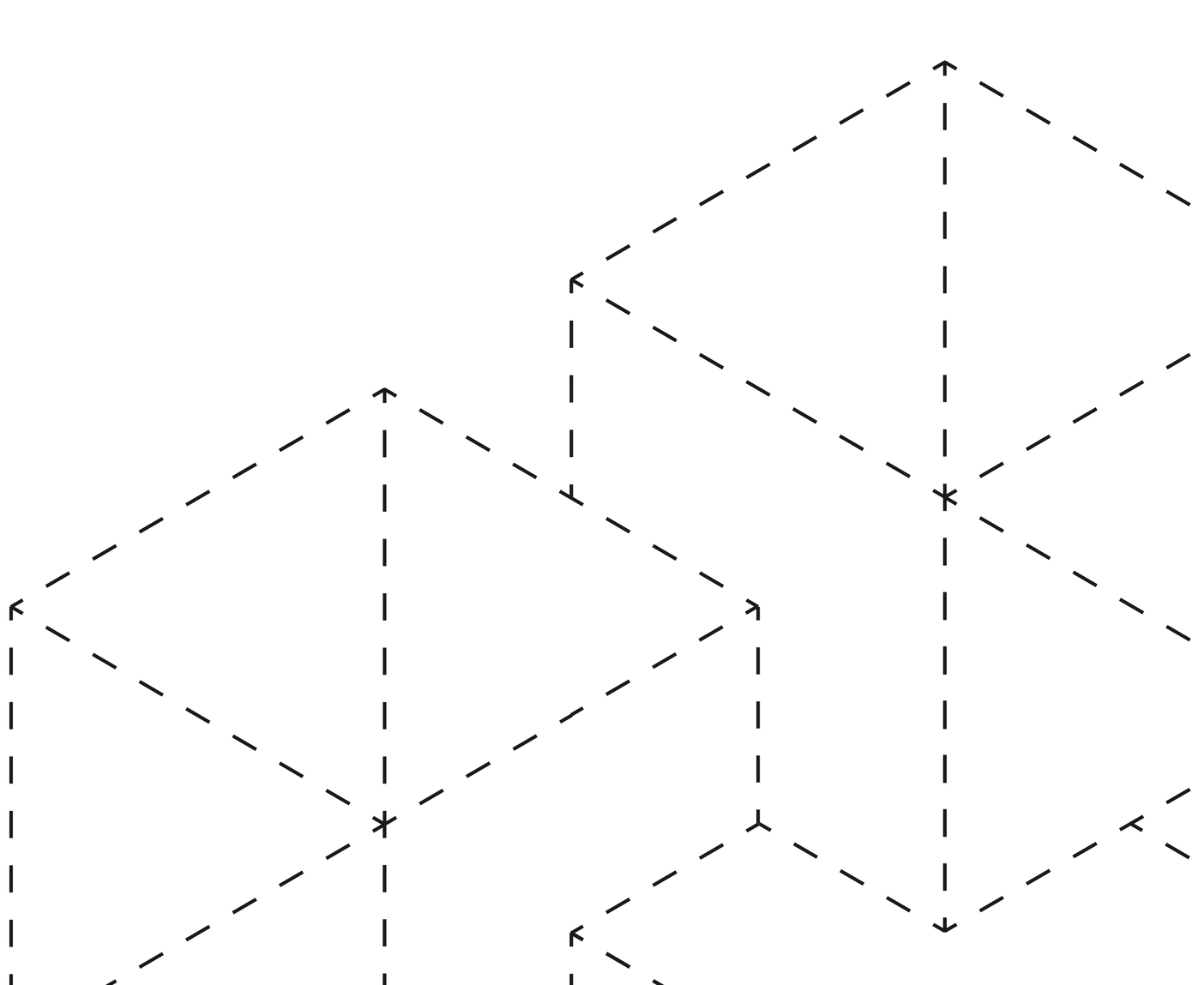
CloudBees resolves this problem by capturing the controller's CasC. This code is maintained as a YAML manifest which is stored within the code base, allowing users to manage their CI solution the same way they manage their infrastructure code (IaC), application code and jobs with a GitOps methodology. This codified version within CloudBees means that software factories can trace a clear auditable history within their CI/CD solution.

These configurations can also be used as modular, portable installs which can be provisioned and redistributed across teams and environments. This additional flexibility allows software factories to pre-configure and offer a leading CI/CD platform from a catalog of configurations defined as YAML. Teams can onboard rapidly and confidently thanks to the production-grade definitions captured by CloudBees CasC.

Observability is a measurable property of a physical system characterized as its ability of being observed. GitOps advocates systems to be designed in such a way that they are easily able to be observed, explored and understood. Git commits related to infrastructure, applications and tools allow organizations to have continuous observability over their current system configurations, upcoming changes and changelogs, thereby reflecting a clear auditable path of what's occured.

Along with observability, controllability represents a major concept of modern control system theory. In order to be able to do whatever we want with a dynamic system under controlled input, the system must be controllable.

Observability goes a long way in enabling us to make sound decisions on how to control the system better. This includes the observability of your CI/CD solutioning. Managing this as code means that a software factory can both observe and control the past, current and future state of the system.



2

Supply Chain Security

Supply chain security is the part of supply chain management that focuses on the risk management of external suppliers, vendors, logistics and transportation.

Its goal is to identify, analyze and mitigate the risks inherent in working with other organizations as part of a supply chain.

The software supply chain is an increasingly complex and muddled landscape. More evolved development strategies such as microservices, feature branches and automation to increase velocity have made the security of this process difficult.

Emphasis on the supply chain's security has never been more important, especially as we look back on one of the largest security breaches in modern computing – see SolarWinds 2021. As the software supply chain has transformed, so must our strategy to secure it. The security is shifting from purely perimeter defense to building security and trust into the supply chain itself.

The software factory should have a method for generating evidence – to include composition, quality and security attributes. A best practice is to create "audit-ready pipelines" that generate the required evidence throughout the CI/CD process.

Why is this important for software factories?

A trusted software supply chain is critical to securely delivering capabilities. An audit trail of components and actions becomes vital to ensuring trust. All government deliverables from development contractors should include both the software bill of materials (SBOM) and the development quality assurance and testing processes and results. The SBOM provides the evidence of what is running where – all versions, third-party libraries and open source components. The evidence should be readily and continuously available to whomever needs it and whenever it's needed (e.g., to pass an audit painlessly).

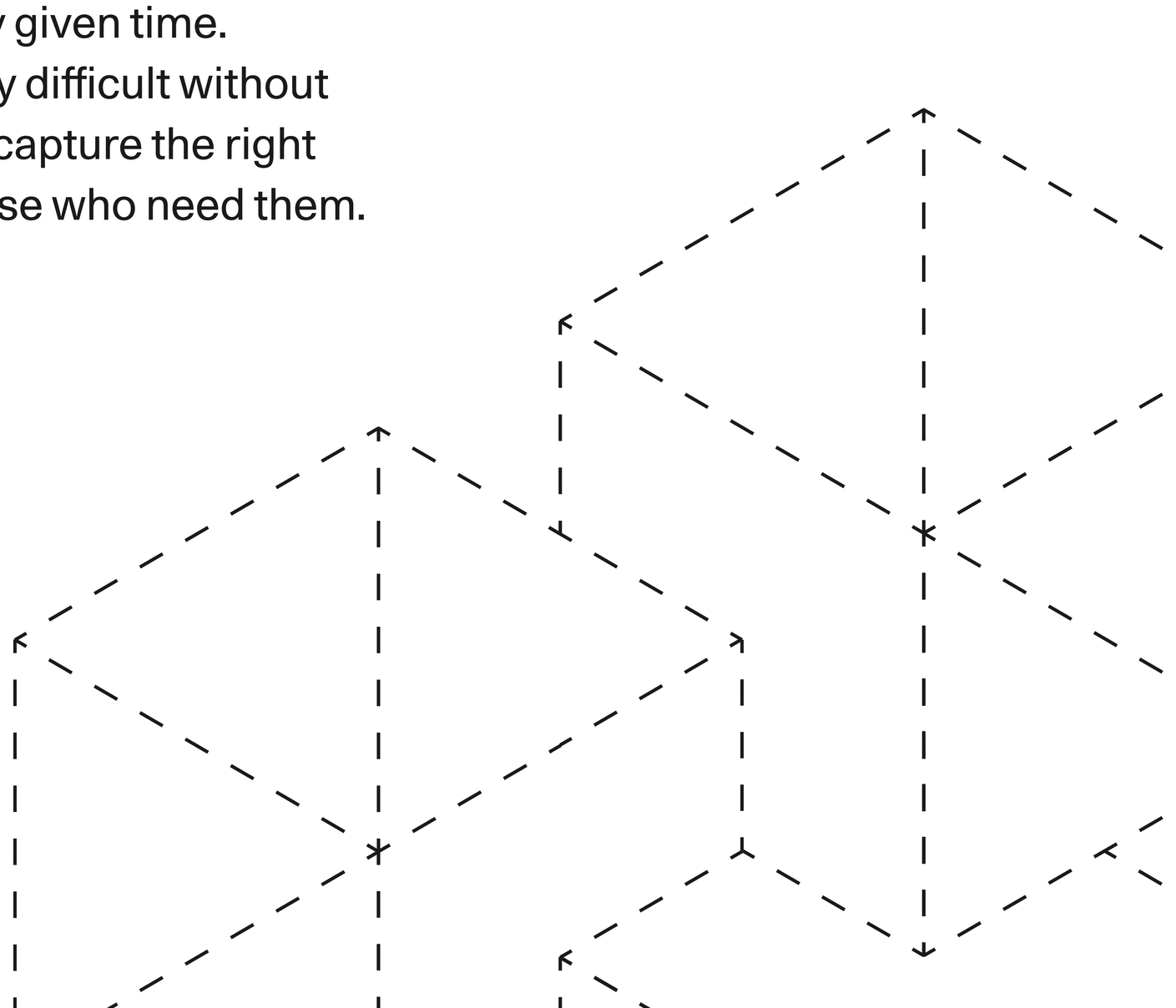
A typical audit process is lengthy and heavy on manual tasks – it is a hunt to recreate all the steps across disparate teams and tools. To conduct an efficient, accurate audit, assessors need visibility across the organization – a detailed understanding of the path that code takes from idea to production. Without that, it is very hard to explain how things have progressed. This kind of evidence collection can and should be automated and continuous. Agencies ought to employ technology that automatically collects evidence and audit data during the execution of every pipeline and stores it in an accessible database for run insights, dashboarding and satisfying compliance goals.

The secure software supply chain is the foundation for continuous Authority to Operate (cATO) as the software delivery process is documented and repeatable – running the process (executing the pipeline) is by definition compliant with the program's and Authorizing Official's (AO) standards.

A secure supply chain achieves the goals of both faster and secure (compliant) software delivery, with less burden on developers, operators, program managers and AOs.

A trusted software supply chain goes beyond the security and integrity of the software being delivered to harden the delivery process itself (the pipeline) as well as to have rehearsed and hardened processes in place to rapidly (or even instantly) respond to vulnerabilities in production. The code is protected in development by ensuring that security scans, dependency checking and other security measures are conducted and passed prior to promoting the code to the next stage. The software delivery process is secured from tampering, unauthorized access or unintended drift. The code and the environments it runs in are also secured during production and constantly monitored with resiliency and instant mitigation built in and frequently tested.

In order to manage and secure the supply chains across the many supported groups in a software factory, there needs to be clear declarative definitions of how software components are built, tested and delivered. Visibility into this process is paramount. A software factory needs the ability to observe and verify the worthiness of any component within the supply chain at any given time. Unsurprisingly, achieving this is extremely difficult without automation and a platform that is able to capture the right data and surface the right insights for those who need them.



How can CloudBees help?

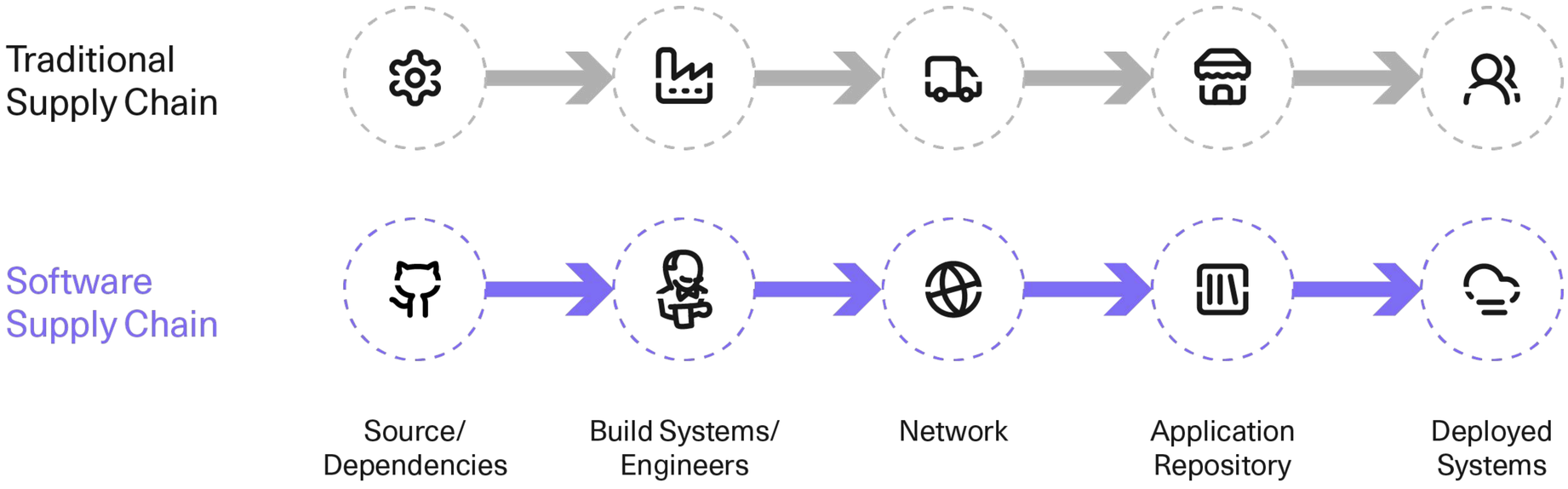
To gauge the increasing attention to software supply chain security, CloudBees conducted an online Global C-Suite Security Survey of 500 C-suite executives from companies across the U.S., U.K., Germany and France. Although the executives expressed overall confidence in the security of their current software supply chains, the survey also revealed the ongoing challenges they face when dealing with emerging attacks.

Security in the supply chain can be a huge undertaking. It's no wonder almost three-quarters of C-suite executives would rather deal with a natural disaster than with a security issue in their software.

While CloudBees knows there's certainly a lot to consider when strengthening your supply chain, it can help to think of this process like an infinity loop, where security is baked into development, delivery and production (and where building, testing, releasing, deploying, operating, monitoring, planning and coding are also part of the equation). Securing every curve and crossover in this loop ensures your company is fortified against disaster.

CloudBees Platform builds trust and confidence into this "loop" by enforcing security, compliance and best practices at every level within the DevSecOps cycle. It begins with the CI process, the way code is injected and initially built within the supply chain. CloudBees also empowers software factories to build and distribute repeatable pipeline definitions.

These pipeline definitions are tested to ensure that pipeline output is both secure and compliant with the organization's standards and policies. The more security and standardization enforced at the top of the funnel, the more efficient the entire supply chain becomes (see State of DevOps Report).



Although this practice should result in production-ready results, there is oftentimes much more work to be done. As software development has become more segmented with the evolution of features-based development and microservices, the path to production has in turn become more complex.

An artifact's path to production is very rarely a clear linear path. Instead, it is reliant on many intersecting paths, dependencies and environments. A software factory needs direct insights across all these intersecting paths. Until this is observable and proven to be compliant, the supply chain cannot be "secure" and fully trusted.

CloudBees Platform works as a Release Orchestrator, which grants observability within the supply chain. More specifically, CloudBees automates the collection of data across the tools that produce release candidates. This data is aggregated in the form of a "body of evidence" and contains the information required to either validate or fail a release. This continuous insight across the supply chain mitigates the risk associated with production releases, gives direct insight into any malpractice within the supply chain and provides an automated delivery loop to resolve any issues.

Crucially, CloudBees enables software factories to securely support and define their own best practices across their own unique toolchains. This is achieved through multiple layers of compliance beginning with the developer CI tasks and extending through observable release strategies.

3

Automating the Release Process

Software factories should have a demonstrable, predictable and repeatable method for release automation that governs the process and configuration of software deployments. Whereas CI aims for speed, CD and release orchestration (CDRO) aim to ensure the “worthiness” of software and getting it where it needs to go.

If CI is analogous to the conveyor belt rapidly producing widgets, CDRO picks up those widgets, validates that they were built properly, ensures they don’t conflict with other widgets, puts them in the right boxes, puts those boxes onto the right trucks, and drives those trucks to the right destinations. CDRO is perhaps the most unsung discipline within software delivery, despite its profound importance.

Why is this important for software factories?

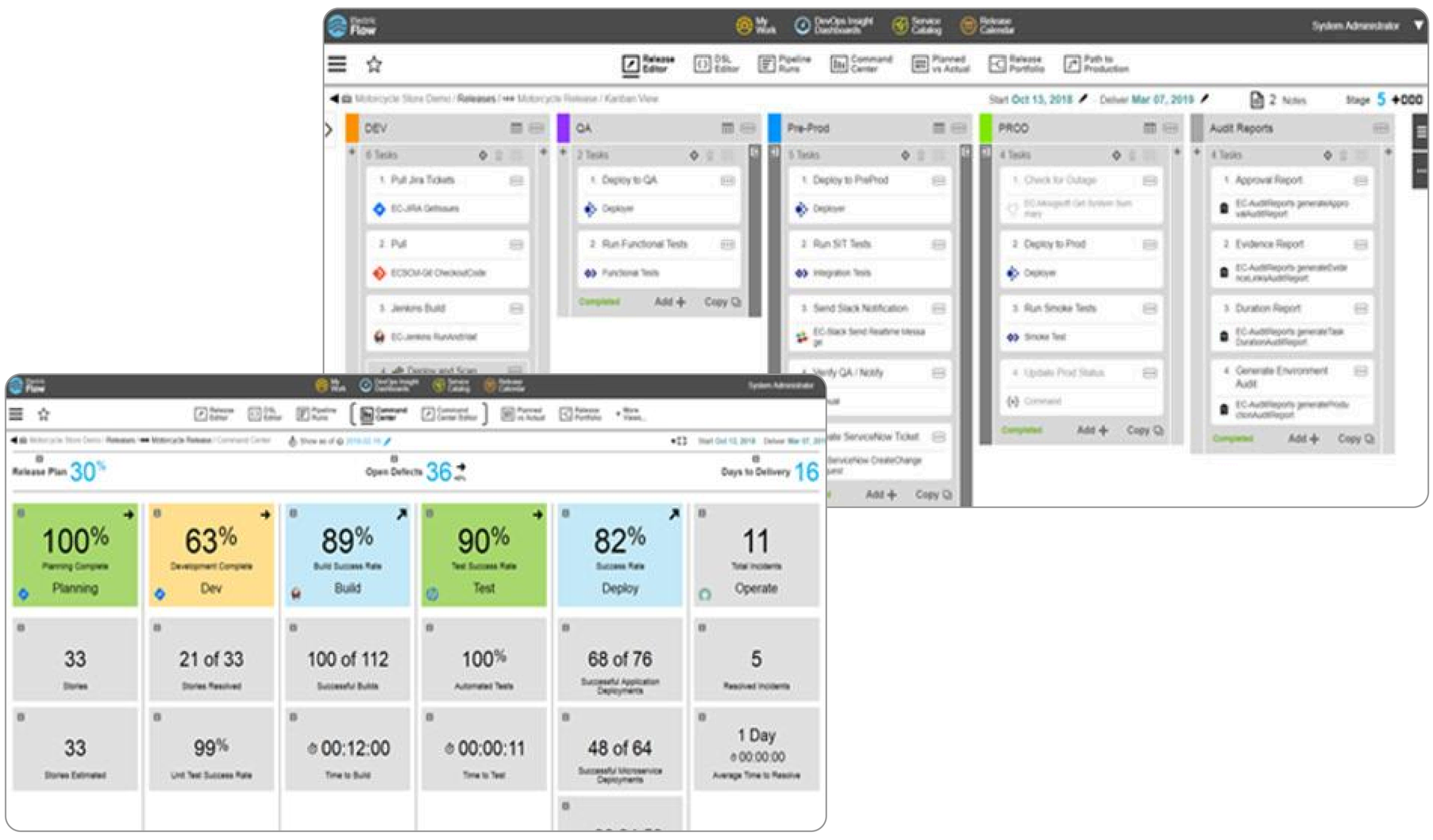
Government programs with substantial software delivery in scope should have teams focused exclusively on accomplishing CDRO. The kind of automation needed to reduce long-term O&M on such programs rarely happens at the task order or scrum team level. (At best, you will achieve islands of automation.) As the program grows, the sprawl of disconnected teams rapidly balloons program costs and threatens the governance benefits of DevSecOps. The following are several elements of effective CDRO:

- Pre-built release templates where automated, customized approval gates control the pipeline, ensuring that each stage is reviewed by the appropriate channel before proceeding.
- Developers working within a connected toolchain should use only immutable objects, which allows the pipeline to become a single point for configuration control and an easy-to-follow audit trail.
- Managers should be able to quickly see the status of all releases across the program. They should have insight and control over releases along with multiple ways to enforce corporate governance, security and compliance.
- Self-service templates of standardized environments and pipelines, accessible for each team and its respective release environment. Each operational environment should get a template that development teams can use, ensuring transition to operations is successful.
- Developers and release managers should easily be able to produce reports for the most common types of audit requests (e.g., the durations of builds, delays in a given release, or an accounting of every manual interaction with a release)

How can CloudBees help?

The CDRO within CloudBees Platform automates and orchestrates complex software releases, pipelines and deployments, providing the analytics and insights to measure, track and improve performance of the software factory.

CDRO is the critical enabler of modern software delivery since its automation can accommodate things like person-in-the loop control gates and testing outside of the CI process. Without it, the release process lacks rigor, consistency and governance, making management painful. The CloudBees Platform's CDRO capability is recognized as the market-leading technology by Gartner and Forrester.



4

Automating the Release Process

Key performance indicators and management insights that let program leadership track progress and make sense of software releases in a consistent, reliable, and repeatable way do not design themselves on the fly.

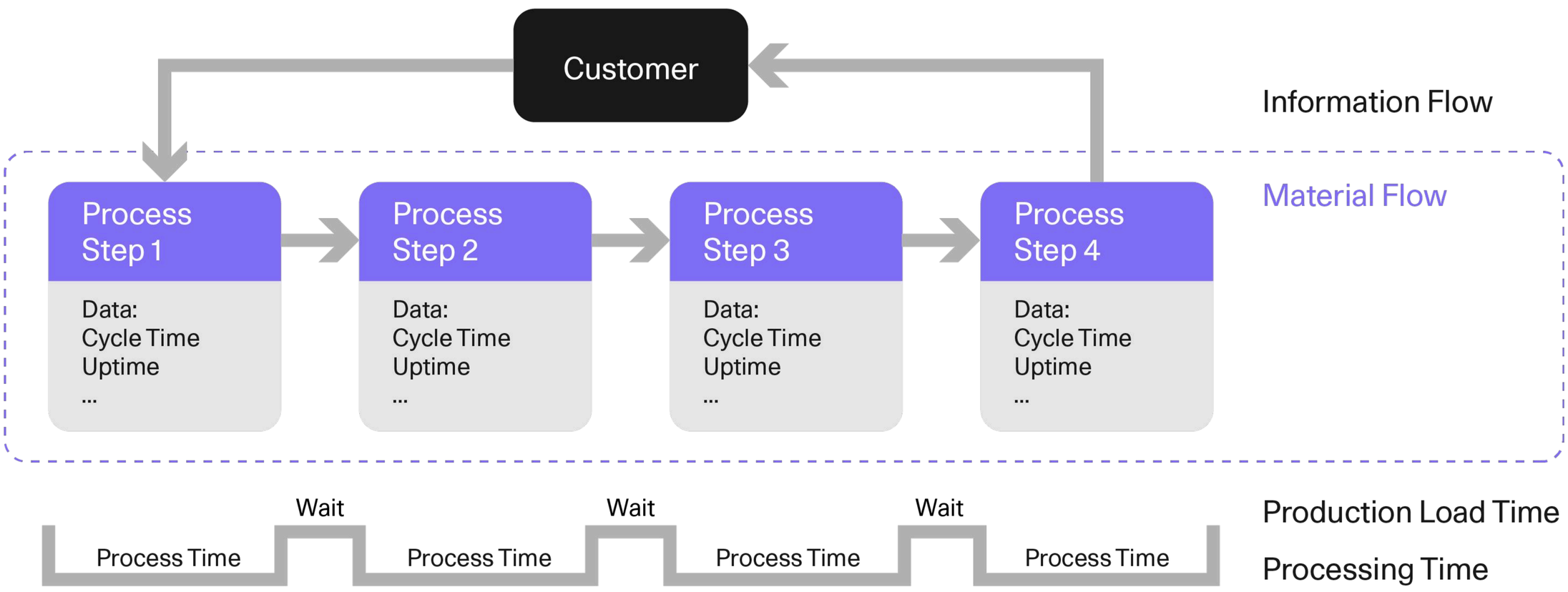
Fortunately, DORA has articulated four metrics that have become industry standard: Mean Lead Time, Deployment Frequency, Mean Time to Recover, and Change Failure Rate. These metrics assess throughput, developer productivity and reliability, which together illustrate the performance of the “software factory.” In its State of DevOps Report, DORA found that the highest performing organizations achieve remarkable results: 46 times more frequent code deployment, 440 times faster lead time from commit to deploy, 96 times faster mean time to recover from downtime, and 5 times lower change failure rate.

Value streams are the steps an organization takes to provide a continuous flow of value to its customers/stakeholders. The goal underpinning value stream management (VSM) is to optimize software delivery by revealing where blockages and DevOps waste occur. This is especially important in large complex programs where waste can disappear into the day-to-day activity. We know that much of the waste occurs in the handoffs (or wait time) between individuals and teams.

Inefficient handoffs lead to low productivity and poor quality; they also inflate the cost of delay. If, however, you monitor performance using VSM, the software factory can align the entire organization around the goal of delivering value.

Software delivery metrics come in many shapes and sizes but traditionally in the DevSecOps world they are guided by DORA, FLOW, and/or ISO standards. A best practice is to track value streams and metrics at both the program and individual team levels. Of course, metrics are worthless if they aren't used to create improvement (i.e., enhance the value stream).

Managing value streams lets you focus on delivering value to the customer. To do this, you need to connect multiple teams, tools and applications to gain clear visibility and insight into how the value is flowing through the software delivery process. This means having a platform that scales easily and enables collaboration, while reducing risk, along with accessing real-time data on DevSecOps performance across teams and tracking throughput and stability.



Why is this important for software factories?

VSM and sensible metrics enable software factories to gauge performance and prove improvements. They can gain real-time performance data to benchmark and track performance based on industry standard indicators related to Throughput (Deployment Frequency, Mean Lead Time) and Stability (Mean Time To Recover, Change Failure Rate).

They can help solve these challenges:

- Poor visibility across the software delivery process makes it difficult to measure and manage DevOps performance – to identify improvements, leverage best practices and support collaboration across teams.
- Lack of information to track flow of value from idea to production, and guesswork on identifying bottlenecks and wait times, leading to increased waste and delays in delivering value.
- No connection between applications, teams and tools means intensive, error-prone methods of collecting static data for measuring and managing the organization's continuous delivery capabilities.

Without insights into the value stream, a software factory has no clear path to increase DevSecOps maturity or modernize and optimize its way of building and delivering software to stakeholders.

How can CloudBees help?

When tracked correctly, metrics reveal extremely valuable data on software factory performance. CloudBees Platform's VSM gives you immediate transparency into your end-to-end software delivery process to measure DevOps performance, remove bottlenecks and improve both velocity and quality. We regularly assist customers with business value assessments. Data from over 250 real-world implementations showed significant hours saved by adopting modern software development practices (i.e., less time spent on unplanned work and rework, and more time spent writing new code and deploying new features). The assessments revealed an average efficiency gain of 66 hours per developer per year. For a 100-person team, this efficiency equates to 6,600 more hours to invest in innovation.

It's one thing for a single team within an organization to perform modern software delivery, but it's an entirely different kind of organization that can achieve the same set of best practices across a complex, multi-team program. CloudBees Platform ties directly into every team's CI/CD workflows and captures metrics which are tied back to DORA and FLOW.

The Platform also pulls data from these workflows in order to generate custom dashboards and reports to reflect any specific metrics a team or manager requires.

5

Hybrid Cloud/IT Modernization/Digital Transformation

IT modernization is the key to reaping the full long-term value and benefits of running your applications and IT infrastructure in the cloud. That means continuous evaluation of your cloud applications, infrastructure and services to ensure they are optimized to achieve your mission and business goals. Continuous modernization results in limited technical debt and the focus shifts from maintaining what you currently have to delivering additional value to your customers.

Why is this important for software factories?

Programs look to software factories for guidance, support and modernization. Almost always this occurs because the programs themselves either lack expertise, resources or time to achieve their modernization goals. Those goals take many shapes including, but not limited to, cloud migration, DevSecOps maturation and adoption of new technologies.

Software factories are not afforded the luxury of offloading this burden; instead, they are charged with accelerating each and every program's path of digital transformation.

The audience they need to support can be large and diverse. This demand creates a need for solutions that can support modern container/K8s strategies alongside mainframe modernization strategies. The harsh reality facing the future of government software factories is that one size most certainly will not fit all.

How can CloudBees help?

As digital transformation scales up in larger organizations, specifically government program offices and large government integration contracts, digital transformation faces a daunting set of institutional and human challenges.

It requires new models of acquisition management, new models of organizational communication and new methods of using technology to manage people. Oftentimes, these challenges include the requirement of systems to be both cloud and traditional on-prem.

CloudBees recognizes this challenge and modernizes the way you deliver value to both modern and legacy systems. These on-prem systems serve as the backbone to many critical services within the government. Too often they are forgotten because it cannot be migrated to the cloud. That is the wrong approach.

Software factories need solutions that both facilitate migration to the cloud and modernize legacy systems that can never exist within the cloud. CloudBees Platform operates within this space of disparate systems in order to drive value through automation and observability within the delivery life cycle.

At CloudBees, we know that continuous delivery best suits organizations that are equipped with a collaborative DevOps culture. In a continuous paradigm where the end-to-end process from idea to deployment is optimized, you cannot afford silos and handoffs between development and operations teams. Companies that don't embrace a DevOps-based culture and fail to knock down silos have difficulty building the kinds of IT and development environments that are required to compete in the digital age.

Learn more

- Watch the Video
  Connecting Software Delivery to Business Outcomes
- Read the eBook
  The Definitive Guide to Modern Software Delivery
- Explore the Case Study
  Internal Revenue Service Modernizes with Agency-wide DevOps Initiative

Learn more cloudbees.com/get-started

CloudBees, Inc.

4 North Second Street, Suite 1270
San Jose, CA 95113
United States
cloudbees.com
info@cloudbees.com