CloudBees

Putting Cloud Native in Perspective

What is it, what can it do for you, and do you *really* need it?

Unless you've been running your enterprise from under a rock, you know that "cloud native" is all the rage. Whatever you're doing, it had better be cloud native! Or so everyone seems to say. But is that *really* the case? As with most things, the truth is less straightforward than what conventional wisdom would have you believe. So buckle up: It's time to dig in, separate the reality from the hype, and find out for yourself how cloud native fits into the larger cloud-migration world.

What Does Cloud Native Even Mean?

If you ask a search engine for a definition of *cloud native*, chances are you'll get gobbledygook. There's a reason for this: Cloud native sells, so there's incentive to keep the definition vague—but expansive. This way the label can be slapped on just about anything. We're not going to do that. For the purposes of this guide, apps can be divided into these broad categories ***based on where they're deployed***.

On Prem (or *Traditional*)

Apps built to run from hardware you own, operate, and maintain in your facilities.

Hybrid

Apps that include on-prem and cloud-hosted (i.e., installed on hardware located in remote data centers) components.

Cloud Adapted

Apps originally built for on-prem deployments that have been migrated to the cloud. These tools can be subdivided into:

Lifted & shifted—On-prem assets that have been redeployed as-is to the cloud, typically via Infrastructure-as-a-Service (IaaS).

Refactored or rebuilt—On-prem assets that have been redeployed to the cloud and updated or otherwise modified to better interoperate with the cloud.

Cloud Native

Tools, apps, or services built from the ground up to operate exclusively from the cloud. Cloud-native assets have only ever existed in the cloud, and are engineered to maximize the advantages of the cloud while minimizing potential downsides.

Pinning Down Terms

Cloud migration has given birth to an endless array of terms that are often conflated with "cloud native." Here are some definitions for cloud-related terms that are ***not*** automatically cloud native:

***Moving to the cloud* (or *cloud migration*)**

The process of transferring IT resources from on-premises hardware and infrastructure to cloud-based services located in remote data centers. These phrases are used to describe both the global IT trend, and hosting scenarios for specific apps.

***Self-managed* (or *self-hosted*)**

Software that is installed and maintained—whether on prem or in the cloud—by the customer rather than the vendor. Apps in this category offer greater control and flexibility at the cost of higher operational overhead.

SaaS (Software as a Service)

A software distribution model where apps are hosted by a service provider and made available to users over the internet. Users typically pay a subscription fee to benefit from regular product updates, security patches, and support services. Because ease of use is a hallmark of SaaS products, this is the largest "as a Service" sector.* Examples include heavyweights like Google Workspace, Salesforce, and Dropbox.

IaaS (Infrastructure as a Service)

A service provider offers virtualized computing resources over the internet. Users rent servers, storage, and networking hardware as well as the virtualization layer—freeing them from the need to buy, install, and manage their own physical infrastructure. Examples include Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). IaaS is the fastest-growing "as a Service" sector,* likely because it offers greater control than SaaS and PaaS alternatives.

PaaS (Platform as a Service)

A service provider offers a platform where users can develop, run, and manage their own apps without the need to maintain the underlying infrastructure. The platform typically includes operating systems, middleware, development tools, database management, and more. Examples include Google App Engine and Heroku. Though PaaS offerings have been criticized for their inflexibility and potential for vendor lock-in, their growth rate is nevertheless second only to IaaS.*

"As a Service" Does *Not* Mean Cloud Native

SaaS, IaaS, and PaaS offerings are perhaps the worst offenders when it comes to cloudwashing. Whether a given service is *cloud adapted* or *cloud native* depends entirely on its development history.

* Source: https://www.gartner.com/en/newsroom/press-releases/2023-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-600-billion-in-2023

Pinning Down Terms

Cloud migration has given birth to an endless array of terms that are often conflated with "cloud native." Here are some definitions for cloud-related terms that are ***not*** automatically cloud native:

Moving to the cloud **(or *cloud migration*)**

The process of transferring IT resources from on-premises hardware and infrastructure to cloud-based services located in remote data centers.

Self-managed **(or *self-hosted*)**

Software that is installed and maintained—whether on prem or in the cloud—by the customer rather than the vendor. Apps in this category offer greater control and flexibility at the cost of higher operational overhead.

SaaS (Software as a Service)

A software distribution model where apps are hosted by a service provider and made available to users over the internet. Users typically pay a subscription fee to benefit from regular product updates, security patches, and support services. Examples include heavyweights like Google Workspace, Salesforce, and Dropbox.

PaaS (Platform as a Service)

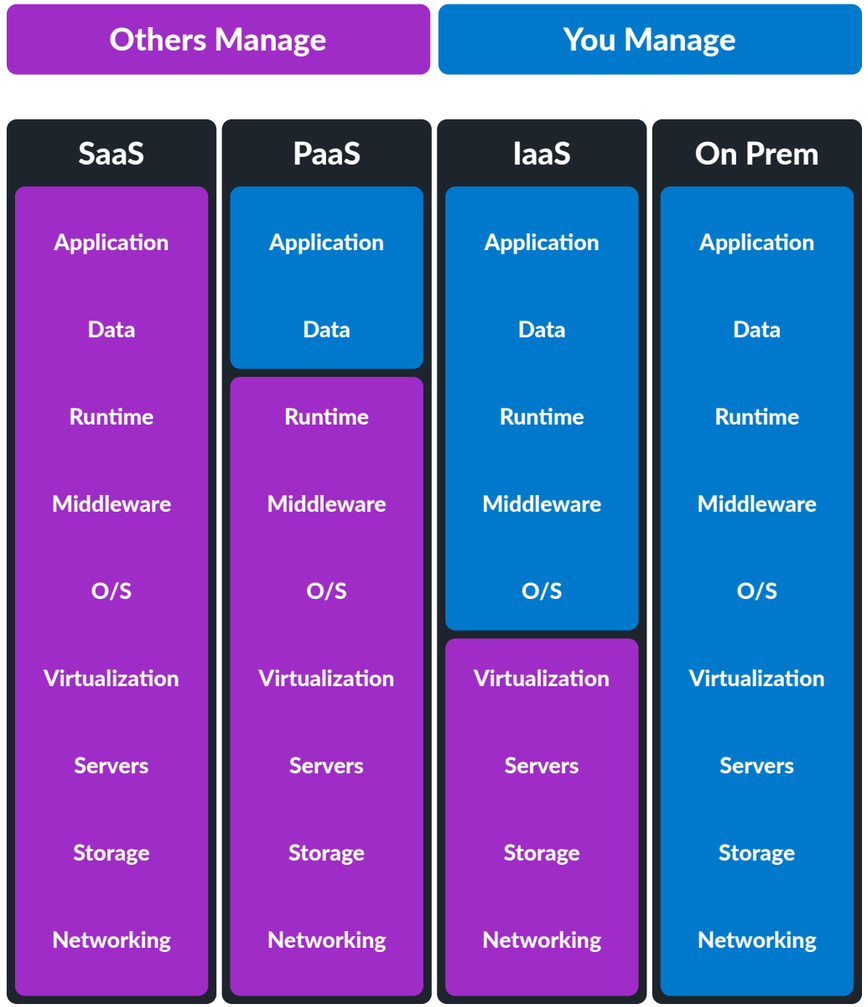
A service provider offers a platform where users can develop, run, and manage their own apps without the need to maintain the underlying infrastructure. The platform typically includes operating systems, middleware, development tools, database management, and more. Examples include Google App Engine and Heroku.

IaaS (Infrastructure as a Service)

A service provider offers virtualized computing resources over the internet. Users rent servers, storage, and networking hardware as well as the virtualization layer—freeing them from the need to buy, install, and manage their own physical infrastructure. Examples include Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP).

Who Manages What?

The choice between SaaS, PaaS, IaaS, and on prem often boils down to how much you prefer to self-manage. Here's how the options compare:



CloudBees.

Identifying Cloud-Native Apps

Since the "cloud native" label is far more common than cloud-native apps, how do you distinguish the *truly* cloud native from the many imposters? Look for the technologies and methodologies engineers use to make cloud-native apps flexible, scalable, and resilient in ways that aren't possible with traditional deployments.

Architecture & Design Principles

Microservices architecture—Organizing apps as collections of small, independent modules that work together to form the complete application. This allows each microservice to be developed, deployed, scaled, and updated independently, providing greater flexibility and resilience than traditional architectures.

API-driven communication—Using APIs for communication between microservices. This facilitates loose coupling and the ability to use different technology stacks for different services based on their individual requirements.

Container & Service Management

Containerization—Packaging apps in containers, which bundle the app along with its related configuration files, libraries, and dependencies into a single unit. This ensures the app will run the same way, irrespective of the environment it's in.

Dynamic orchestration—Managing the complexity of numerous interconnected services and containers through use of an orchestration platform, like Kubernetes, that takes care of scheduling, load balancing, and distribution of containers across the system.

Service meshes—Using tools like Istio and Linkerd to improve interoperation and communication across microservices.

Tools & Platforms

Cloud-native tools—Harnessing tools that are foundational to cloud-native workflows, like Kubernetes, Docker, and Amazon's Elastic Container Search (ECS).

Cloud-Native Technologies

Elasticity/ephemerality—Leveraging microservices and containerization to take advantage of on-demand resource allocation. Resources like storage, compute instances, or containers can be spun up and torn down quickly, often lasting only for the duration of a specific task. This greatly facilitates scaling, resiliency (components are quickly replaced if they fail), and cost efficiencies (there's no need to pay for idle infrastructure).

Optimized resource usage—Taking full advantage of cloud-specific technologies and services, such as serverless computing, managed databases, and artificial intelligence (AI) / machine learning (ML).

Resiliency—Exploiting the distributed nature of cloud-native apps to make them highly tolerant to failure and capable of rapid recovery—often without human intervention. Cloud-native apps often continue running when components fail, and new instances or resources can be rapidly spun up to replace those that have failed.

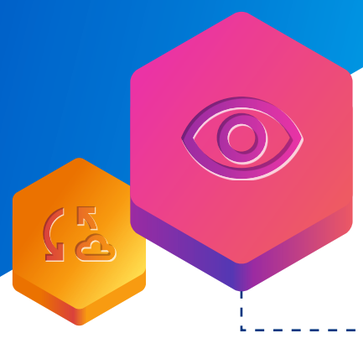
Infrastructure Management

Infrastructure as Code (IaC)—Utilizing IaC, where infrastructure setup is automated and managed using code, to enhance the reproducibility and scalability of supporting infrastructure.

Observability—Emphasizing observability through use of integrated logging, monitoring, and tracing functionalities that describe the system's state and behavior.

Identifying Cloud-Native Apps

Since the "cloud native" label is far more common than cloud-native apps, how do you distinguish the *truly* cloud native from the many imposters? Look for the technologies and methodologies engineers use to make cloud-native apps flexible, scalable, and resilient in ways that aren't possible with traditional deployments.



Architecture & Design Principles

Microservices architecture—Organizing apps as small, independent modules.

API-driven communication—Facilitating communication between microservices and promoting loose coupling.

Container & Service Management

Containerization—Bundling apps in containers with all necessary dependencies.

Dynamic orchestration—Managing the interconnected services and containers.

Service meshes—Using tools like Istio and Linkerd for better microservices communication.

Cloud-Native Technologies

Elasticity/ephemerality—On-demand resource allocation.

Optimized resource usage—Leveraging cloud-specific technologies like serverless computing and managed services.

Resiliency—Ensuring high tolerance to failure and capability of rapid recovery.

Infrastructure Management

Infrastructure as Code (IaC)—Automating infrastructure setup and management.

Observability—Integrating logging, monitoring, and tracing to understand the system's state and behavior.

Tools & Platforms

Cloud-native tools—Harnessing tools that are foundational to cloud-native workflows, such as Kubernetes, Docker, and Amazon's Elastic Container Search (ECS).

Note: These are the same goals engineers adopt when refactoring/rebuilding on-prem apps for the cloud. Cloud-native apps just tend to do these things better than refactored/rebuilt apps because they aren't held back by the need to negotiate original, on-prem codebases or entrenched legacy workflows.

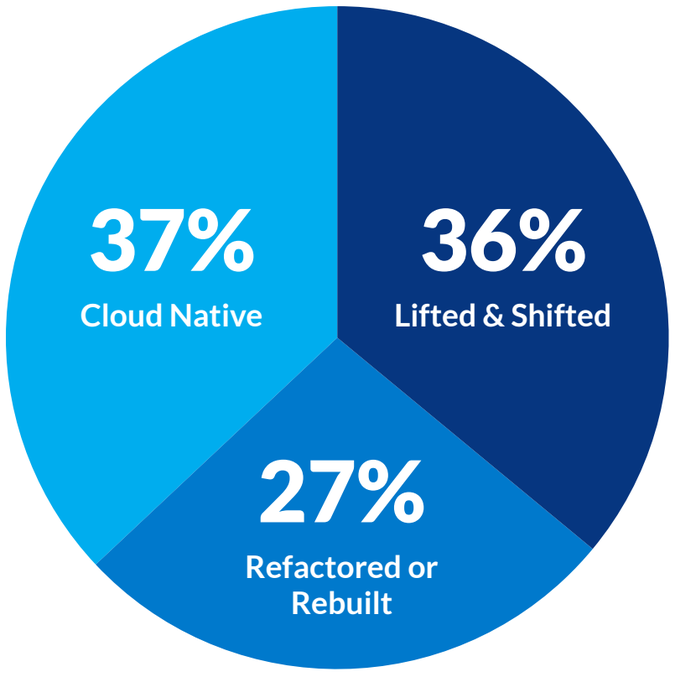
How Critical Is Cloud Native to Enterprise IT?

There's no straightforward answer here. To start, it's important to remember that the value of any tool lies primarily in how well it's designed—not where it's deployed. So a great on-prem app is almost always better than a poorly designed cloud-native app. Who cares if an app is *containerized* if the app isn't any good, right? And this scenario is not uncommon; tried-and-tested tools are more likely to be on prem, as cloud-native upstarts that have yet to demonstrate their long-term value.

That said, the cloud (native or otherwise) is critical because it's where enterprise IT is heading … albeit *slowly*. It's easy to get the impression that the shift toward the cloud is tsunami-like, but this is hardly the case. Consider this: Gartner looked at IT spending *only in enterprise IT categories that can transition to cloud*: application software, infrastructure software, business process services, and system infrastructure markets. Even within this tailored subset of the larger IT world they predict *only 51% of IT spending will be cloud-focused by 2025*.* So, here's **Takeaway #1: The death of on prem has been greatly exaggerated**.

Even if we focus on the portion of IT that is already cloud focused, cloud-native apps have a way to go before they dominate:

Primary Method of Application Deployment to the Cloud**



In summation, a *subset* of the apps that have moved to the cloud within the *subset* of enterprise IT categories that *can* be moved to the cloud are cloud native. And we now have **Takeaway #2: Cloud native isn't the be-all-end-all of enterprise IT or the cloud-migration story**.

The bottom line is that cloud-native apps represent a fraction of the tools used in IT today. While the trend line toward cloud native is clear (for now; see the *Trends are Just Trends* sidebar), whether a specific app should be cloud native will always depend on the user, the industry, and both the economic and technological implications of building that app for the cloud.

* Source: https://www.gartner.com/en/newsroom/press-releases/2022-02-09-gartner-says-more-than-half-of-enterprise-it-spending

** Source: 2023 State of Cloud Native Security Report

Trends Are Just Trends

While there's a lot of excitement surrounding cloud migration, bear in mind that the cloud is primarily an *economic proposition*: the idea that it's cheaper to rent datacenter resources than build them.

While this is often true—startups rarely have the capital, time, or personnel to deal with infrastructure—things get murkier as enterprises scale. If you have millions of active users, building your own infrastructure may be less expensive in the long run than paying giant datacenter bills in perpetuity. For this reason, some enterprises eventually bring certain assets back to on-prem hardware.

Keeping your options open is a better gameplan than running headlong toward an exclusively cloud-native future.

How Critical Is Cloud Native to Your SDLC?

This requires a two-part answer, because your SDLC represents two things: the collection of tools you use to build your apps, and your apps themselves—the products you ultimately deliver to your customers.

Your tools—Cloud-native tools are not automatically the best tools—in fact, some are downright awful. For this reason, as your teams continue to seek the best tool for any task, they will rely on some combination of on-prem, hybrid, cloud-adapted, and cloud-native tools for years to come. To give them the freedom and flexibility they'll need to pursue any best-in-class solution—wherever it lives—you should prioritize architecting an SDLC that allows tools across these environments to interoperate harmoniously.

Your apps—Whether your products should be cloud native is, of course, between you and your customers—if cloud native matters to them, it should matter to you. That said, whether you're already pursuing cloud-native products or only thinking about them, you should have the capability to deploy cloud-native apps.

This brings us to **Takeaway #3: You don't need cloud-native tools to make cloud-native apps**. How you build your apps and where you deploy your apps are two distinct things!

Fitting Cloud Native into Your Cloud Migration Strategy

Most organizations will continue to use a mix of on-prem, cloud-adapted, and cloud-native digital assets for the foreseeable future. The variables will be how these assets are distributed across their IT landscapes, how that distribution changes over time, and how smart they are about leveraging the unique strengths of these tools while avoiding their pitfalls. With that in mind, here's a quick primer covering the essential pros and cons of each:

There are persuasive reasons for enterprises (both yours and your customers') to continue using tools across all deployment scenarios. While cloud native is an important piece of the larger cloud-migration puzzle, **your application portfolio should be focused on three objectives:**

1. Being able to integrate ***any*** application—regardless of where it's deployed—into your SDLC with minimal friction.
2. Being able to use ***all*** of your applications harmoniously no matter where they live—even as your stack evolves.
3. Being able to deploy your products ***anywhere*** your customers might want them.

Centering on these goals ensures that your DevOps teams are unencumbered by architectural limitations and that your products can adapt in real time to shifting economic, technological, market, and user demands.

Pros & Cons of Deployment Scenarios

On Prem	Hybrid	Lifted & Shifted	Refactored/Rebuilt	Cloud Native
PROS				
✓ Complete control over infrastructure ✓ High level of security ✓ Customizable environment	✓ Highly flexible & customizable ✓ Can be cost-efficient by using the best environment for any task ✓ Improved scalability over on prem	✓ Easy to implement ✓ Quick cloud migration ✓ No immediate need to redesign applications	✓ Better optimization than lift & shift ✓ More cloud benefits without fully rebuilding application ✓ Balance of cost & performance	✓ High scalability ✓ High resilience ✓ Can be cost-efficient, depending on scale ✓ Innovation-friendly
CONS				
✗ High upfront costs ✗ Maintenance & upgrade responsibilities ✗ Lowest scalability	✗ High complexity ✗ Requires careful planning & management ✗ Can increase security concerns due to inter-connectedness of systems	✗ Doesn't fully utilize cloud benefits ✗ Costs can be high due to non-optimized applications ✗ Legacy system issues may persist	✗ Requires substantial time & resources ✗ May need significant changes in application architecture ✗ Potential service interruptions during redesign	✗ Requires skill set for cloud-native development ✗ Complexity due to microservices ✗ May require organizational changes ✗ Initial setup may take time ✗ Dependency management

CloudBees.

Which Tools in Your SDLC Should Be Cloud Native?

Deciding whether a given tool in your SDLC would benefit from being in the cloud—whether that means lifted-and-shifted, refactored/rebuilt, or cloud native—is a case-by-case thing. Zeroing in on cloud native, several questions come to mind: Is there a cloud-native option available? If so, is it high quality? You might be surprised how often the answer to one of these questions is "no." Assuming you've found something worth looking into, here are factors worth considering:

Suitability—If developers are facing challenges related to scalability, flexibility, deployment speed, or working with distributed teams, migration to cloud-native tools can help. The first step is to understand what isn't working well with current tools and processes. Next, consider whether those pain points would be best served by a cloud-native solution—or if you'll create as many problems as you solve.

Training & skill set—Transitioning to cloud-native tools may require upskilling. Developers may need to learn new technologies and approaches, such as working with microservices, containers, or infrastructure as code. Do you have masters of Kubernetes standing by? Teams should consider whether they have the necessary skills. If not, are they ready to invest in acquiring them?

Costs—Cloud-native tools *can* offer cost savings in terms of operational efficiency, but there are often costs associated with the transition, including training, tooling, and potentially higher operational expenses for cloud services. It's important to do a comprehensive cost analysis.

Security & compliance—Cloud-native tools can offer robust security features, but they also present unique security challenges. If your organization has strict data security, data sovereignty, or other compliance requirements, you need to ensure that the cloud-native tools you use meet those requirements.

Tool compatibility—Your cloud-native tools should be compatible with each other—and with any existing tools that will continue to be used. Interoperability can save a lot of headaches down the line.

Availability & disaster recovery—Cloud-native tools hosted by reputable providers generally offer high availability and robust disaster recovery options. However, you should review these capabilities to ensure they meet your organization's needs.

Support & community—Consider the level of support the tool vendor provides. A strong, active community can also be a valuable source of help and resources.

Vendor lock-in—Some cloud-native tools may lock you into a particular cloud provider. Consider how easy it would be to move to a different tool or provider if your needs change in the future.

Deciding to go cloud native should be based on a thorough analysis of your enterprise's and your application's specific needs, capabilities, and strategic objectives. Remember, not every tool in your SDLC needs to be cloud native, and a mixed-environment approach—seamlessly blending on-prem, hybrid, cloud-adapted, and cloud-native tools—will likely remain best for many years to come. This yields **Takeaway #4: Don't focus on where a tool is deployed; focus on using the best tools in the best ways**.

Evaluating Cloud-Native Tools for Your SDLC

Determining if a tool in your Software Development Life Cycle (SDLC) should transition to a cloud-native approach is nuanced and requires careful consideration. When contemplating cloud-native alternatives, consider the following key factors:

Tool availability and quality—Before moving forward, ascertain if there's a high-quality cloud-native version of the tool you're using. Not all tools have cloud-native counterparts that meet industry standards.

Suitability—Pinpoint current limitations or challenges in your SDLC. Cloud-native tools can enhance scalability, flexibility, and deployment velocity while supporting distributed teams. Begin by identifying shortcomings in existing tools and evaluate if a cloud-native solution can alleviate these pain points without introducing new ones.

Training and skill set—Transitioning often mandates a new skill set. Familiarity with microservices, containers, or infrastructure as code could be vital. Determine if your team possesses these skills or is prepared for training.

Cost analysis—While cloud-native tools can streamline operations, transitioning might entail expenses such as training, new tools, or potentially greater cloud-service charges. It's imperative to perform a thorough cost-benefit analysis.

Security & compliance—Cloud-native solutions can offer enhanced security. However, they may also introduce distinct security considerations. Ensure any tool under consideration adheres to your organization's data security and compliance mandates.

Tool compatibility—Ensure that cloud-native tools are not only compatible with each other but also with legacy tools that remain in use. Interoperability can prevent future integration challenges.

Reliability, availability, & disaster recovery—Reputed cloud-native tool providers typically guarantee high availability and robust disaster recovery measures. Ensure these measures align with your organizational demands.

Support & community—Consider the level of support the tool vendor provides. Additionally, a strong, active community can be a valuable source of help and resources.

Vendor lock-in—It's essential to assess how tied a tool is to a specific cloud provider. Gauge the ease of migration to a different tool or provider should organizational needs evolve.

Deciding to go cloud native should be based on a thorough analysis of your enterprise's and your application's specific needs, capabilities, and strategic objectives. Remember, not every tool in your SDLC needs to be cloud native, and a mixed-environment approach—seamlessly blending on-prem, hybrid, cloud-adapted, and cloud-native tools—will likely remain best for many years to come. This yields **Takeaway #4: Don't focus on where a tool is deployed; focus on using the best tools in the best ways.**

Building Your Bridge to the Future

If a theme has emerged, it's that—when we strip away the hype—cloud native is a still-small piece of the much larger enterprise IT story. We'll all be dealing with the full spectrum of deployment scenarios for the foreseeable future. For developers seeking to thrive in the present while ensuring their place in the future, the question is, ***what should we be doing right now?*** The answer is pretty straightforward:

1. Build an SDLC that allows you to use any tool ***in any environment***.
2. Make sure that SDLC facilitates deploying your apps ***to any environment***.
3. Leverage your SDLC's flexibility to use the tools—and deliver the apps—that are best suited to ***your enterprise's business objectives***.

The future might be all-cloud-native-all-the-time, but we're not there yet. For now, flexibility, efficiency, and ease of use matter infinitely more than where your tools are deployed. So try to remember **Takeaway #5: Having the power to use any tool to build any app is what really matters**.

CloudBees®

CloudBees—On Prem to Cloud Native, and Everything in Between

Architecting the SDLC described here is a tall order. If you'd like help building your bridge to the future, the CloudBees Platform is worth checking out. It's a collection of DevOps tools that operate in concert to bring visibility, harmony, and flexibility to your SDLC. Highlights include:

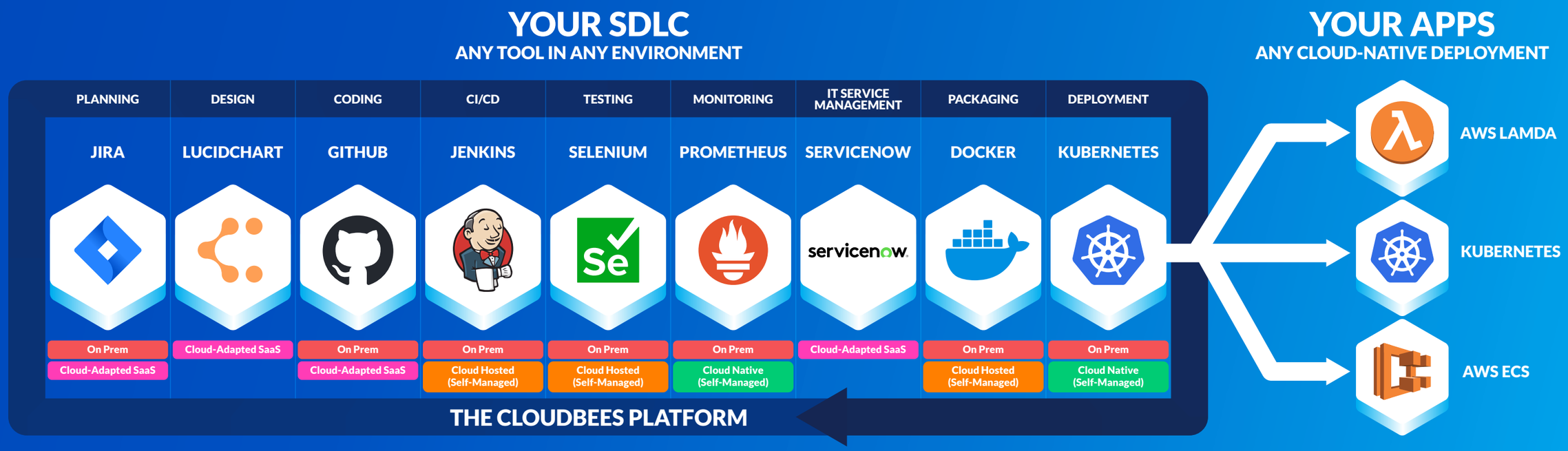
- **Centralized visibility & control—**single-pane oversight and management of all SDLC tools/processes, regardless of environment
- **CI/CD—**built on an enterprise-grade Jenkins implementation (on prem or cloud hosted)
- **Release orchestration—**orchestration of any tool, for any app, across any environment
- **Progressive delivery via feature management—**supporting canary releases, blue-green deployments, and A/B testing
- **Continuous security & compliance—**everything you need to assess, assert, and evidence security and compliance across your entire SDLC

CloudBees—Any Tool, Any Environment, Any Deployment

The CloudBees Platform offers a high-ROI toolkit designed to help you use any tool to deploy apps *anywhere—including cloud-native environments such as AWS ECS, AWS Lambda, Kubernetes, etc.* The illustration below shows what a typical CloudBees pipeline might look like, demonstrating how your SDLC can leverage a variety of tools to ultimately deliver cloud-native experiences to your customers.

CloudBees®

Reach out to us to discuss how the CloudBees Platform can transform your SDLC *and* your business.



CloudBees®