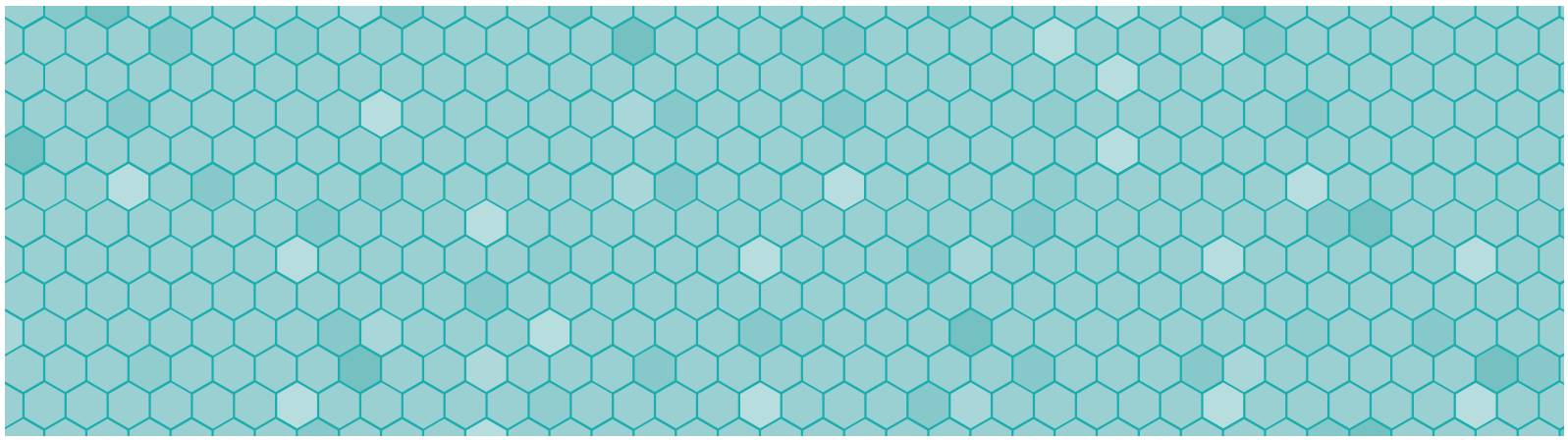

Continuous Cloud Delivery:

Delivering Better Software, Faster, Continuously. In the Cloud.

The emergence of ubiquitously available public cloud resources continues to have a profound effect on software development and delivery approaches.

Continuous Delivery – a development and delivery methodology where you keep your application in a release-ready state at all times – is rapidly becoming less of a goal for companies and more of a business imperative.

The CloudBees Platform provides all the pieces you need to deliver better software, faster, all the time. You get the resources you need in the cloud, when you need them, shared across teams or isolated, as required. Builds, tests and release rollouts are tracked end-to-end, letting you get product to market more quickly, with reduced risk. You can connect to your existing systems and work with your existing investments, so continuous delivery can provide returns immediately. We call this Continuous Cloud Delivery, and you can get started today.

CloudBees

The Java™ Paas Company

Table of Contents

Overview

If your business depends on reaching customers and partners over the Internet, you are facing a major set of challenges today, such as:

- Expectations on uptime and responsiveness are higher than ever. Hosted services and mobile apps are in use 24x7, and downtime is costly in terms of money, customers and credibility.
- Consumer experience with mobile and hosted services has conditioned people to expect a constant drumbeat of feature/function delivery, not the yearlong release cycles, installation headaches and upgrades of the past.
- There is relentless demand to evolve and build on your existing software investments more quickly. Many software projects today depend on connecting users to valuable backend software systems.
- You need better information to help gauge the trade-offs between expensive technology investment, time-to-market and revenue. Nimble startups are using rapidly delivered micro-features to guide decisions on further software investments, minimizing costs and lowering risk.

The tools you have traditionally used to develop and deliver software are not aging well in the brave new world of the cloud. You've stretched the value of virtualization technologies to the limit within your data center. Your traditional vendors all have dollar signs in their eyes as they dress up tired, old packaged software in cloud clothing and parade it in front of you. If you're already delivering hosted services to your users and partners, then you also know that the constraints inherited from last-gen packaged software delivery and QA models are holding you back.

If you're a developer, you probably already have good ideas about how to respond to these challenges from a technology and methodology perspective. Your tools of choice may include Jenkins Continuous Integration (CI), automated testing and agile practices of various kinds, and you may already be using them effectively. Even so, getting to the next level – continuous delivery – can be difficult because of barriers like compute resource shortages, long setup times, problems hosting and maintaining infrastructure shared between teams.

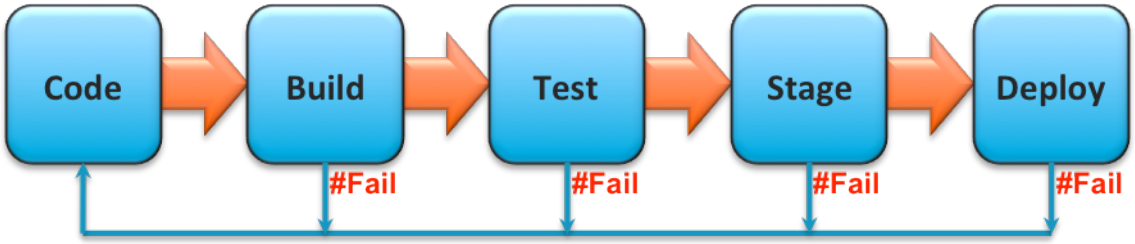

Figure 1 – Continuous Cloud Delivery Overview Requires a Complete Develop/Test/Stage/Deploy Platform

Continuous Cloud Delivery is aimed squarely at helping developers and operations solve these problems. It combines the philosophy and mechanics of continuous delivery with the power of the CloudBees Platform as a Service to get you past many of the common problems and to a working solution quickly. You can take advantage of continuous delivery today,

using the CloudBees Platform. In this paper, we will examine the benefits of Continuous Cloud Delivery, what it takes to implement it in your organization and how to get started.

Benefits

Continuous Cloud Delivery brings measurable benefit to your business. Let's take a look at four ways it impacts your bottom line.

Time to Market

Time to market is, without a doubt, the key driver for PaaS adoption. Continuous Cloud Delivery helps you create new products and services more quickly than ever before because you have the resources you need, when you need them. For example, using Continuous Cloud Delivery, Choose Digital (http://www.cloudbees.com/case-study/choose-digital.cb) was able to develop their branded digital marketplace as much as five times faster than if they had developed it using a traditional process.

Continuous Cloud Delivery eliminates:

- Delays waiting for IT to procure machines
- Arguing over whether you really need resources to get your job done
- Sharing resources between teams and developers, that often results in delays and errors
- Creation of mocked resources that inadequately represent production interactions and result in late-binding surprises in a release cycle
- Custom assembly and maintenance of the development and testing tools you need to get your job done

Eliminating delays and surprises caused by resource shortages and workarounds are only one part of the way Continuous Cloud Delivery improves time to market. Perhaps more importantly, Continuous Cloud Delivery helps reduce the "big bang" approach to new software delivery. No modern development project emerges fully formed, but many projects are plagued by long integration and stabilization periods, while relatively autonomous teams reluctantly step up to the cost of dealing with the parallel streams of development done by others. Continuous Cloud Delivery lets this integration and testing process happen naturally, in real time, at very low cost to each team, thereby incentivizing everyone to avoid costly backend integration activities. The result is that complex software deliveries are working earlier and released faster.

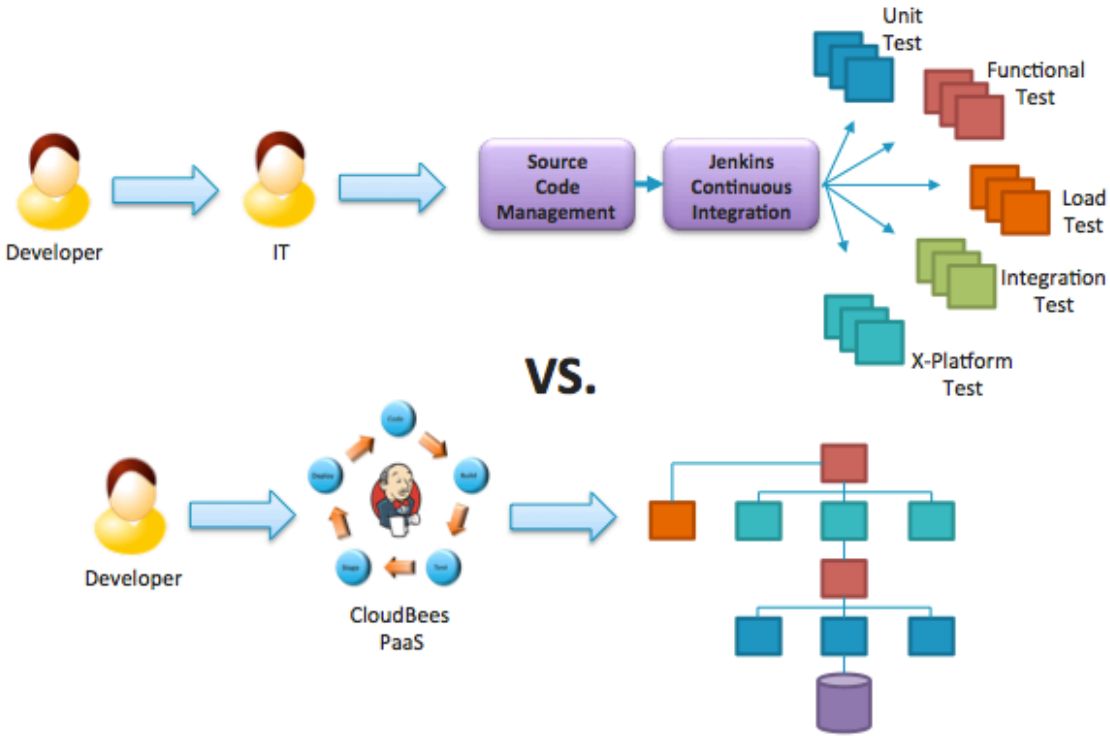

Figure 2 – Continuous Cloud Delivery Makes More Efficient Use of Resources

Reduced Risk

Development teams use the tools they have available, and those tools shape the way the teams work. When you suffer from scarce resources, you compensate by cutting corners, time-boxing access between teams and other techniques to "do more with less." Without elastic resourcing served up painlessly by PaaS, teams tend to back-load resource-intensive operations, like stress and integration testing. Back-loaded operations make every issue more difficult to trace back to its origin and every verified fix more costly to propagate to completion. This behavior puts your project at risk of slipping release dates. You may be forced to make difficult and ill-informed tradeoffs on quality and user experience. Or, you will have to cut feature/function to compensate. The iron triangle becomes a way of life with costly consequences. Continuous Cloud Delivery doesn't defy the physics of software development. But, by helping you to maintain and measure quality continuously, while delivering functionality constantly in a working state, the magnitude of risk associated with a release is kept to a minimum. And by making it simple to roll back to a previous state and control the deployment process while monitoring behavior in real-time, releasing quality software can become as simple as committing code or as structured and controlled as you want. A great example of streamlined yet controlled deployments is, LoseIt! (http://www.cloudbees.com/case-study/lose-it.cb). They have used Continuous Cloud Delivery on CloudBees to cut the time to propagate fixes from two weeks to just five minutes.

It's also important to recognize that continuous delivery enables cheap failure and rapid experimentation. When you roll releases out in long cycles, you are by definition making big bets. Continuous delivery lets you make small bets and understand outcomes before betting more, reducing incremental risk and total risk.

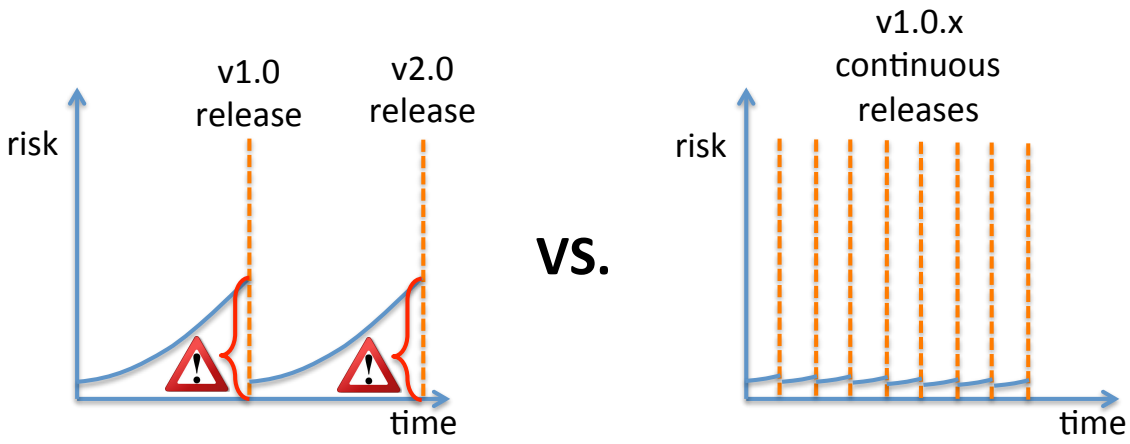

Figure 3 – Risk and Cost Are Reduced with Continuous Delivery

Improved Quality

No tool offers a silver bullet for software quality. Producing high-quality software takes a combination of plain old hard work, a top-down and bottom-up organizational commitment and an investment of real resources. Continuous Cloud Delivery helps out on all of these axes. Continuous integration gives you the machinery to drive automated tests, helping to magnify every investment you make in creating tests and making software more testable by projecting them across your organization. Simply having the ability to deploy the results of your work on an ongoing basis, for consumption by other teams, can be a huge advantage on a large project. The advantage of visibility and share-ability that the cloud brings about are tremendous compared to the more siloed approach present in most organizations. The most common adoption pattern of Jenkins today is a viral one, with new teams hearing about what their colleagues are doing, repurposing a box under someone's desk and then spreading the word. Continuous Cloud Delivery reinforces that viral behavior by making resources available on-demand, while giving a centralized view into the larger organization. The result is better information, better quality and an environment that helps you enforce the quality values that are important to your organization – like code coverage, style, performance metrics and so on – systematically.

Real-time Feedback

When you have a release cycle of a year or more, you spend a lot of time in requirements definition before coding and you work hard to get beta feedback when you deem your work to be feature complete. And of course, there are many modern day variations on the ancient waterfall cycle. However, when you are able to keep your software in a release-ready state at all times from a quality and user experience perspective, you must also adapt the processes you use for gathering

requirements and feedback. Most importantly, you want to let your deliverables do the work for you to measure success and detect failure. Continuous Cloud Delivery helps you do this in a number of ways:

- You can use techniques like A/B testing and dark deployments to gather end-user feedback during production usage of your product or service.
- You can use hosted services to measure and report customer usage and engagement as you work to improve the way customers use your product or service.
- You can correlate product usage metrics, over time, with revenue and adoption metrics, giving you a view into the effectiveness of engineering investment on your bottom line.

Continuous Cloud Delivery enables low-cost continuous experimentation, and with the right tools to gather feedback, translates experiments into revenue opportunities. All of these efforts have another beneficial side effect: developers become more aware of their role in the business, improving their ability to impact revenue and fueling their thinking of new ways to grow the business.

Elements of Continuous Cloud Delivery

CloudBees provides Continuous Cloud Delivery by bringing together a range of key hosted services across the full application lifecycle. The extensibility points in the CloudBees Platform ensure that the most demanding and custom needs of enterprises can be met.

Hosted CI

Jenkins is the most popular continuous integration server in use today. Kohsuke Kawaguchi, a CloudBees employee, originally created Jenkins as an open source project. Jenkins has a tremendously involved and committed community behind it. Jenkins is easily extended using its plugin model, and the community contributes 3-4 new plugins every week. This activity likely means that Jenkins is already integrated with the tools you want to use; and if it's not, then you can integrate it yourself with the help of many others who have done similar work. CloudBees provides Jenkins as a hosted service. This means:

- CloudBees experts are managing Jenkins itself, keeping it up-to-date and making sure it is always up and running.
- You have instant access to a huge pool of compute resources, as our hosted Jenkins service scales-out using as many executors for builds and tests as needed. You get parallelism in test and matrix builds using scale-out cloud resources.
- You only pay for what you're using, as you're using it.
- You have access to the additional value-add of Jenkins Enterprise by CloudBees plugins. These plugins address advanced security, performance and scale-out needs.

Beyond the rich set of Jenkins plugins, our hosted environment also gives you instant access to hosted partner services that can improve code quality (like Sonar) and expand testing capabilities (like SOASTA and Sauce Labs).

Integrated Deployment Process

As you create and advance your applications, you can deploy them immediately to CloudBees RUN@cloud™. RUN@cloud supports Java and JVM-based runtimes, in addition to virtually any language runtime, using our ClickStack™ capability. CloudBees' extensive integrated technology partner ecosystem brings a variety of hosted services together. You can make use of best-of-breed application performance monitoring, log management, mail and database service providers, among others.

Jenkins makes it simple to construct test pipelines and promotion criteria between related efforts. So, you can set up processes that are fully automatic or that are gated by release management or other approval mechanisms you already have in place. Several patterns are used, the most common being to duplicate development systems and a staging system. Thus, as artifacts are produced as a result of developer checkins, those that pass the required tests are always staged in a pre-production environment. These staged bits are guaranteed to be exactly the same ones as have been tested, but the actual push to production is gated by an authorized role using the Role-based Access Control (RBAC) plugin available within Jenkins Enterprise by CloudBees. The owner of that role causes a build to kick off, which then pushes to production.

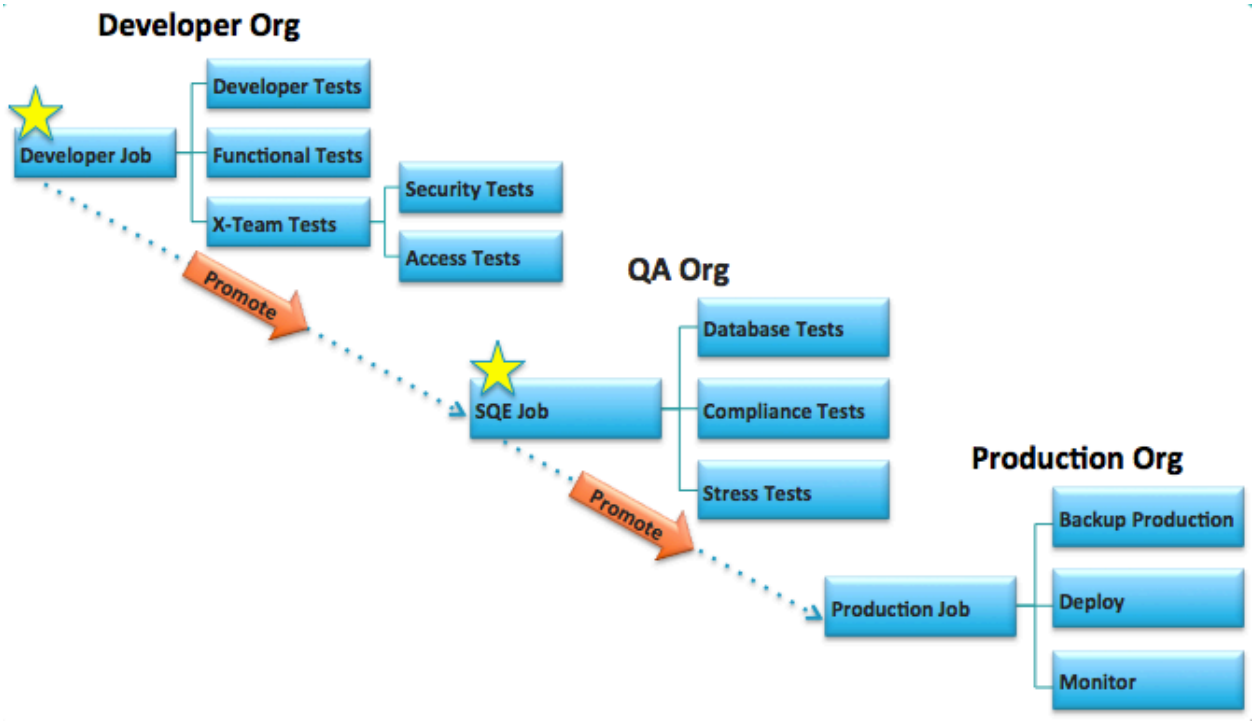

Figure 4 – Pipelines and Promotion Processes Combine for Flexibility and Control

Role-based access control, along with multiple CloudBees accounts partitioned by organization or function, provides the flexibility needed to map to virtually any processes deemed critical by your organization. Jenkins numerous notification mechanisms help you to coordinate across teams and keep the delivery machinery running. Jenkins plugins connect to many other systems of record – source code systems, artifact repositories and so on – making it easy to deploy to many targets, including on-premise systems or alternative PaaS providers like Google App Engine and CloudFoundry.com.

Artifact/Change Tracking

Using the built-in fingerprinting capabilities within Jenkins, every change you make can be tracked from its source origin through testing, staging and production. Jenkins also integrates well with artifact repositories, like JFrog's Artifactory, both on-premise and in CloudBees DEV@cloud™. As a result, CloudBees provides the kind of traceability needed for even the most complex project to push release-ready code into staging and production with confidence that the binaries can easily be tied back to specific code changes.

Even the most straightforward application often interacts with existing systems. These systems might reside within your company or they might take the form of a third-party service, such as are often used in mobile applications. A common architectural practice is to put an intermediary layer of processing between your application and a back-end system. In any of these cases, your stream of development intersects with another stream of development. The intersection is a point of friction in the development and deployment process. You proceed at one pace, while another team proceeds at a different pace. Or, during development, changes made by one team cause another team's code to fail. While continuous integration and delivery, if done correctly, serves to minimize these issues, when they happen, you need complete traceability to sort out the problem.

Rollback

The most extreme practitioners of continuous delivery, like Facebook, operate in an "always roll forward, never roll back" mode. However, for most organizations, having an ironclad guarantee that rollback is possible, simple and reliable is a must. At its simplest, rollback consists of just identifying the version of your application to deploy on RUN@cloud, since CloudBees tracks versions of applications. Deployment and rollback can both be scripted. For complex deployments with multiple dependencies or subsystem interactions, a wealth of cloud resources that you pay for only when you're using them means that you can very cheaply and easily replicate your entire production environment (likely with external references to certain data sources and systems). A blue-green deployment approach then lets you push load off to the new version of a complex system, with the knowledge that your entire existing production system is still in place if you need to rollback. You can then remove and repurpose the old production systems when you are sure everything has been performing properly for some time. This kind of Continuous Cloud Delivery solution is not practical within the constraints of on-premise data centers and licensed software. Aside from cost and risk reduction, the logistics of such a rollout are transformed from weeks to hours.

Other Considerations

Continuous delivery is not a new concept or term. If the advantages are so self-evident, why isn't everyone doing it? The changes enabled by the cloud and PaaS are going to give you a great boost toward your goal, but let's look at some other considerations as you explore what may be a new world for you and your organization.

Non-technical Concerns

The most frequently cited issues in implementing continuous delivery are not technical ones. They are cultural and organizational. Continuous delivery requires a top-down and bottom-up commitment to test automation and to the value of testing overall. Even when one team of an organization "gets" the value, in the end they are often just one team that must live within the constraints of a larger organization. Continuous Cloud Delivery does not solve these problems, but it can help. It helps by making sure your team has the resources it needs without drawn-out, bureaucratic negotiations. It helps by exposing your work to management and other teams more directly. It helps by giving you a way to showcase the value of continuous delivery when your team is meeting its obligations more quickly and with better results than others. And it helps by arming your management with the data it needs to drive change beyond the borders of your team.

Technical Concerns

If you're working on a "green field" project, then you're very much in the minority in IT. Most developers are maintaining or extending existing software and a lot of that software is "hard to test." When you develop software using modern testing techniques, you make sure your code is tested as you create it. You use tools like unit testing, code coverage and test automation frameworks, and these tools are the yin to continuous delivery's yang. If you are tasked with modifying or extending a program written by old-school developers who created a piece of software that was thrown over the wall to a QA team focused on functional and/or manual testing, you have a problem that no amount of cloud-speak will cure. You can start to approach the problem by automating the existing tests; however, if possible, defining clear, testable interfaces to the existing system, and implementing extensions outside of it, is probably the best approach to this kind of project.

Another practical concern of continuous delivery is the investment it takes to create, partition and layer tests in a way that minimizes the useful time-to-feedback. Ideally, every check-in is gated by unit tests that, if successful, kick off more extensive and time-consuming tests. Many tests can be parallelized, so the time between a developer making a mistake, being notified of a problem, and propagating a fix is short. However, creating this type of optimized system takes an investment to get right. Getting the buy-in to improve test suite efficiency can be surprisingly hard if the right team and individual incentives are not in place, so this technical concern is typically tied to the non-technical organizational and cultural ones.

Conclusions and Next Steps

You can see now how Continuous Cloud Delivery helps get better software delivered faster. What's the best way to get going?

Start small. If you're new to Jenkins and continuous integration overall, try a CloudBees ClickStart™ that is aligned with the technology stack you're interested in. That will help you quickly get your head wrapped around the end-to-end dev-test-deploy cycle and connections to hosted services. From there, you can explore setting up pipelines and promotion mechanisms.

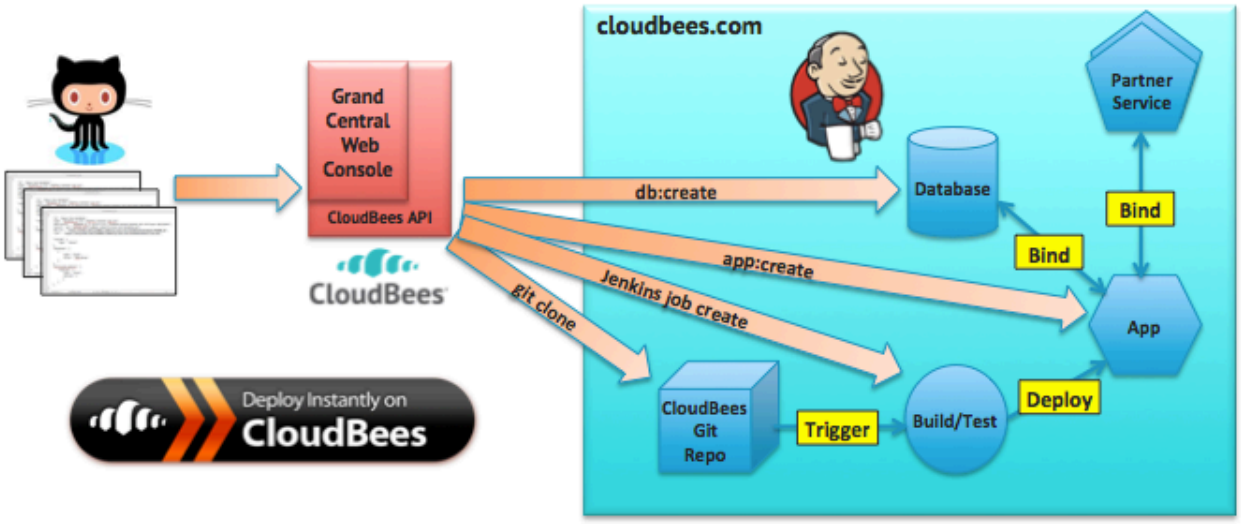

Figure 5 – Use ClickStarts to Get Going Quickly with Your Favorite Technology Stack

If you're already using CI, then come on over to CloudBees and discover how easy it is to make use of parallelism and on-demand resources. Check out our blogs (http://blog.cloudbees.com) that show you the basics and best practices of pipelining and promotion processes. Sign up (http://www.cloudbees.com/signup) to try out the CloudBees Platform for free (no credit card required). Get engaged with our Customer Success Program and let us help you to make the best use of Continuous Cloud Delivery in your business!

CloudBees

CloudBees, Inc.

400 TradeCenter, Suite 4950
Woburn, MA 01801
USA

www.cloudbees.com
info@cloudbees.com