DZone

THE GUIDE TO CONTINUOUS DELIVERY

VOLUME II FEB 2015

BROUGHT TO YOU IN PARTNERSHIP WITH

stackato by ActiveState

ANSIBLE

ca technologies

CloudBees

dynatrace

redgate ingeniously simple

Sauce LABS

snap

DEAR READER,

I am very excited to welcome 2015 with the release of our Guide to Continuous Delivery. 2014 was a huge year for DZone Research with the release of six incredibly informative research guides that have now been downloaded over 150,000 times.

DZone's 2014 Guide to Continuous Delivery was a huge success with over 30,000 downloads and still counting. The enormous amount of positive feedback we received last year from our audience has motivated us to produce another edition of the Continuous Delivery research guide, taking an even further look at DevOps best practices.

In the 2015 edition of the research guide we have expanded the focus into other areas of DevOps implementation and management, including a look at how Continuous Delivery practices seep into other aspects of the organization outside the development team.

We have focused on the role of containerization, automation, testing, and other best practices that move Continuous Delivery beyond simply an exercise in tooling. The solutions directory provides a comprehensive listing of the various ARA, CI, and CM tools beneficial for practitioners in the DevOps space.

We are already looking forward to an exciting year with several interesting new topics that we plan to cover, including Application Performance Management, Software Quality, the Internet of Things, and Big Data/Business Intelligence. We are excited to hear your feedback on this guide, thoughts on future topics, and ways you think we can make our research guides better suited to your needs.

Thanks again for your time and for being a part of our great community. Wish you all a wonderful year ahead and happy reading!

JAYASHREE GOPAL
DIRECTOR OF RESEARCH
research@dzone.com

TABLE OF CONTENTS

CREDITS

DZONE RESEARCH

John Esposito
Editor-in-Chief

Jayashree Gopal
Director of Research

Mitch Pronschinske
Sr. Research Analyst

Benjamin Ball
Research Analyst

Matt Werner
Market Researcher

John Walter
Content Curator

Ryan Spain
Content Curator

Lauren Clapper
Content Curator

MARKETING & SALES

Kellet Atkinson
General Manager

Alex Crafts
VP of Sales

Ashley Slate
Director of Design

Chelsea Bosworth
Marketing Associate

Chris Smith
Production Advisor

CORPORATE MANAGEMENT

Rick Ross
CEO

Matt Schmidt
President & CTO

Brandon Nokes
VP of Sales, Demand Generation

Hernâni Cerqueira
Lead Software Engineer

Special thanks *to our topic experts Andrew Phillips, JP Morgenthal, Jim Bugwadia, Matthew Skelton, Sharone Zitzman, and our trusted DZone Most Valuable Bloggers for all their help and feedback in making this report a great success.*

Summary & Key Takeaways

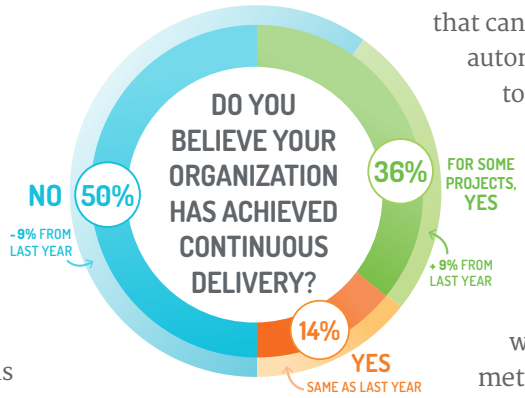
BUSINESSES ARE ALWAYS GOING TO BE LOOKING TO ADOPT software delivery practices that help them come to market on time, on budget, at high quality, and with few to no failures along the way. Nothing about these goals has changed—what's changing is the way we achieve them. Previously this meant a greater focus on the development team, but it goes beyond that, farther even than the operations team and QA, and extends into how the entire organization operates. Today's Continuous Delivery implementation isn't just an exercise in tooling or a list of best practices, it reaches into every element of the business. The DZone *2015 Guide to Continuous Delivery* has more insight than ever into the status of DevOps in the enterprise and the obstacles facing developers, not only in their tooling, but within the organization as a whole.

The resources in this guide includes:

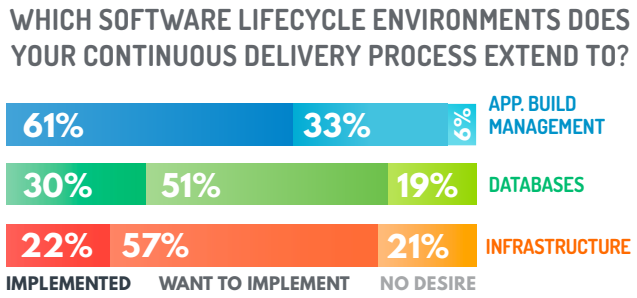
- Side-by-side feature comparison of the best configuration management, continuous integration, and application release automation tools (selected based on several criteria including solution maturity, technical innovativeness, relevance, and data availability).
- Expert knowledge for implementing Continuous Delivery and DevOps best practices
- The tools that developers are using at every stage of the Continuous Delivery pipeline
- Assessing how excellence in testing affects the entire enterprise
- Analysis and forecasting based on research collected from 900+ IT professionals

KEY TAKEAWAYS

CONTINUOUS DELIVERY: IT'S DEFINITELY A GROWING PRACTICE Exactly half (50%) of the 900+ IT professionals we surveyed believe that they have implemented Continuous Delivery for some or all of their projects. This represents an almost 10% growth from last year's research. Our data indicates that while many professionals and organizations might be attempting to implement Continuous Delivery, they often aren't meeting the ideal criteria. When we filtered respondents by stricter criteria based on three key traits in the definition of Continuous Delivery [1], the number dropped to 18% overall. It's important to note that this is a 10% increase over last year—more teams are getting it right.



CONTINUOUS DELIVERY FOR DATABASE MANAGEMENT AND INFRASTRUCTURE While we did see some growth in Continuous Delivery adoption to different software lifecycle environments, we're still waiting for aggressive adoption with database management and infrastructure. Application build management is still the most popular CD environment, with 61% of all respondents reporting its usage. We also saw a 9% growth in the adoption of Continuous Delivery practices for databases—a significant growth for an observation period of less than a year. The growth of database implementation is not surprising considering increasing adoption of Continuous Delivery for database lifecycle management [2]. True Continuous Delivery has to be present at every stage of the software production process.



THE CULTURE OF CONTINUOUS DELIVERY IS AN OBSTACLE WORTH TACKLING Organizational culture plays a major role in achieving Continuous Delivery, but CD can also causes improvements in cultural metrics like collaboration and efficiency. 64% of our audience reported that "corporate culture" and a lack of collaborative practices was the biggest obstacle facing their implementation of Continuous Delivery—and this is an obstacle that can't be overcome with proper tooling and automation. Company culture also matters to IT professionals when it comes to measuring the success of Continuous Delivery: 41% of our respondents reported that "cultural metrics" like team collaboration and organizational efficiency play a role in measuring the impact of Continuous Delivery practices, which made it the third most important metric of success after support ticket frequency (46%) and outage frequency (43%).

[1] http://martinfowler.com/bliki/ContinuousDelivery.html
[2] https://www.simple-talk.com/sql/database-administration/continuous-delivery-for-databases-microservices,-team-structures,-and-conways-law/

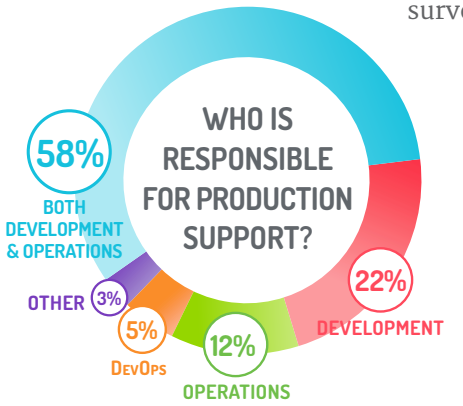
KEY RESEARCH FINDINGS

More than 900 IT professionals responded to DZone's 2015 Continuous Delivery Survey. Here are the demographics for this survey:

- Developers and Engineers made up 65% of the total respondents.
- 62% of respondents come from large organizations (100 or more employees) and 38% come from small organizations (under 100 employees).
- The majority of respondents are headquartered in Europe (48%) or the US (28%).

DIVISION BETWEEN DEV AND OPS IS CLOSING

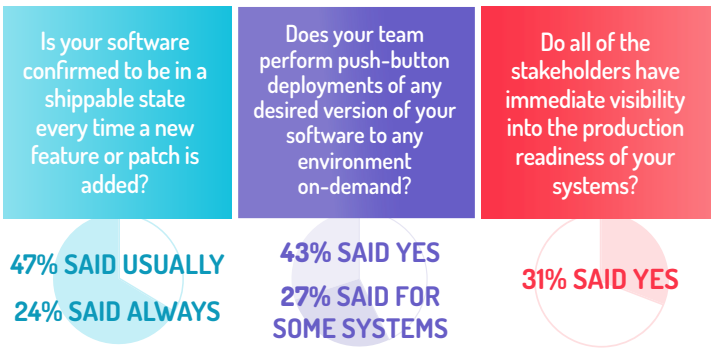
Continuous Delivery has always been positioned as being part of the larger DevOps philosophy. To implement this philosophy, many companies choose to create a team that is focused on cross-compatible skills for multiple disciplines between development and operations. 35% of the survey respondents have an officially designated DevOps team (up 5% from last year). For organizations without DevOps teams, the division of labor between development and operations teams can become blurry.



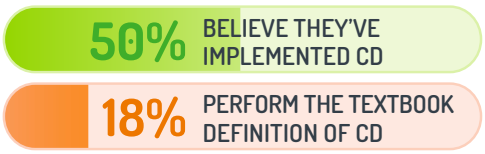
For example, development teams were only slightly more likely to perform code deployments to production (45%) than operations teams (32%). 58% of respondents that said both development and operations were both responsible for production support.

MORE PROFESSIONALS ARE ACHIEVING CONTINUOUS DELIVERY

The authors of the Continuous Delivery methodology defined three key traits to determine when an organization has fully implemented its practices [1]. The panels below shows how many survey respondents have these traits in their systems:

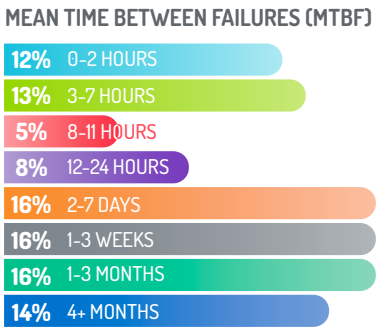


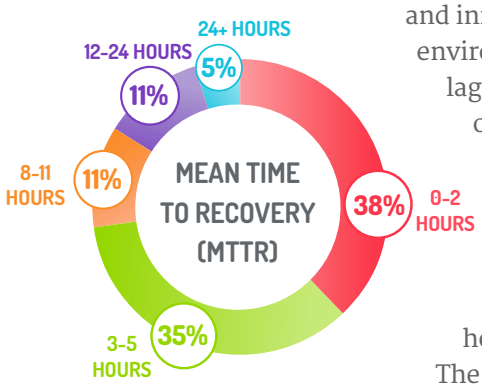
Most organizations are somewhere in the process of having adopted Continuous Delivery practices, but not having fully achieved the primary principles. 36% of those surveyed believe they have achieved CD for some of their projects, and 14% believe they are doing it in all of their projects. The combined statistic is that 50% of respondents have implemented Continuous Delivery for some or all of their projects, which represents a 9% growth in implementation over the last year. To determine the amount of respondents performing a "textbook" implementation, respondents were filtered by the three key traits of Continuous Delivery from the section above; 18% said "yes" or "always" for all of the questions. While this is smaller than the half of respondents who believe they've implemented Continuous Delivery, this number still represents a 10% growth compared to the number of respondents who had achieved "textbook" Continuous Delivery last year (8%).



QUICK TO RECOVER AND SLOW TO FAIL

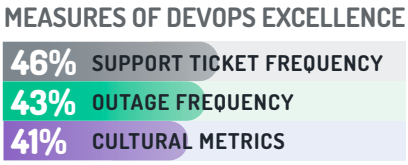
Among other deployment and support metrics, we checked into the Mean Time to Recover (MTTR) and Mean Time Between Failure (MTBF) for our respondents' support and operations teams. The survey indicated that recovery time after a failure (MTTR) averaged close to 6 hours. The respondents also reported that the average time between failures (MTBF) is just over 4 hours, with 13% reporting.





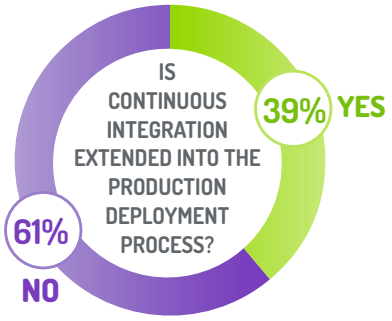
CULTURE IS BOTH OBSTACLE AND MEASURE OF SUCCESS

The three biggest barriers to Continuous Delivery adoption are company culture (64%), a lack of time (63%), and team skillsets (45%). This is the second time that company culture and a lack of time have topped the list of reasons why IT professionals are having problems adopting CD practices. The healthy growth of certain practices within a company culture has always been a major focus within DevOps, so it seems natural that negative culture would be a barrier to implementation. It's not surprising then that the responsiveness of DevOps culture can also be used to measure the success of Continuous Delivery. Cultural metrics (41%) are the third most important measure of success after support ticket frequency (46%) and outage frequency (43%).



CONTINUOUS DELIVERY IS SPREADING TO OTHER ENVIRONMENTS

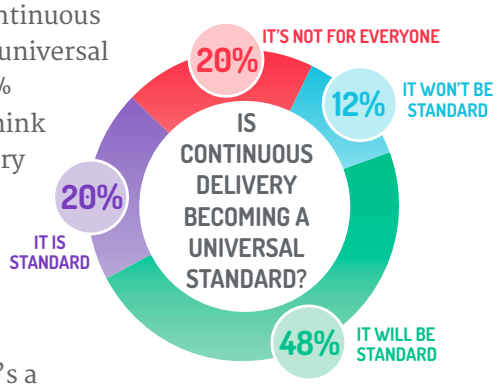
Continuous Delivery isn't a hard sell for developers. 61% say that they have already implemented it for their application build environments, and only 6% have no desire to do so. Database and infrastructure environments are still lagging behind, but they've seen decent growth over the last year. 30% of respondents have implemented it for database environments and 22% have implemented it for infrastructure, which represents a combined 13% growth in these environments. Another positive sign is that over half of all respondents hope to implement CD for those environments.



The survey results do show some promising stats for organizations taking the first steps toward complete CD. 39% of respondents say they have extended their CI practices to production deployments.

CONTINUOUS DELIVERY IS BECOMING THE UNIVERSAL STANDARD

As popular as Continuous Delivery has become, it would be a stretch to say it's currently a universal standard for software delivery, though it's not that hard to say that it's well on its way. Respondents largely praised Continuous Delivery as a near universal standard, with 20% saying that they think Continuous Delivery is currently a universal standard, and 48% saying it will soon become the standard; that's a combined 68% of respondents. Even the negative responses were relatively tame—20% of respondents just don't think CD practices are appropriate for every environment. 12% said they just don't think it'll be a universal standard.



[1] http://martinfowler.com/bliki/ContinuousDelivery.html

CONTINUOUS DELIVERY POWERED BY JENKINS

CLOUDBEES

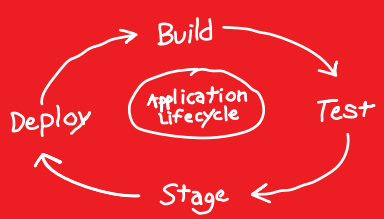
THE ENTERPRISE JENKINS COMPANY

Jenkins at Enterprise Scale

- *Scalability*
- *Manageability*
- *High Availability*
- *Support*

Jenkins Your WAY

- *On-Premise*
- *Cloud*
- *Hybrid*



Jenkins for Continuous Delivery

- *Process Modeling with workflow*
- *Checkpoints for easy restarts*
- *Analytics*

CloudBees
The Enterprise Jenkins Company

info@cloudbees.com
www.cloudbees.com

Pre-Requisites for a Successful Enterprise Continuous Delivery Implementation

Businesses today are moving toward continuous delivery as a methodology to meet the ever-increasing demand to deliver better software faster. Continuous delivery can be seen as a natural evolution from continuous integration and agile software development practices, but the cultural and operational changes to achieve it are great. While these changes are a huge challenge, there are many practical steps you can take to make real progress today. **Here are six:**

1. **Make sure the development, QA and operations teams all have shared goals and that they communicate.**
2. **Continuous integration is a prerequisite for continuous delivery – get CI right, first.**
3. **Automate and version everything**
4. **Share tools and procedures between teams**
5. **Make your application and infrastructure both production- and project-friendly:** deployments should be non-events - all the tooling and documentation required to empower the development and QA teams and make them autonomous should be provided.

> "Continuous delivery can be seen as a natural evolution from continuous integration and agile software development practices."

6. **Make application versions ready to be shipped into production.** Continuous delivery is not just about a set of tools, ultimately it is also about the people and organisational culture. Technology, people and process all have to be aligned. If organisations are to reap the rewards of a more fluid, automated approach to software development that can also provide them business agility – they need to implement these best practice steps on the path to continuous delivery.

WRITTEN BY
Cyrille Le Clerc, *Director, Product Management , Cloudbees*

Choosing CD Tools That Bring Your Team Together

BY MATTHEW SKELTON

WITH AN EVER-INCREASING ARRAY OF TOOLS AND TECHNOLOGIES claiming to enable Continuous Delivery, how do you know which tools to try or to choose? In-house, open source, or commercial? Ruby or shell? Dedicated or plugins? Highly collaborative practices such as DevOps and Continuous Delivery require new ways of assessing tools and technologies in order to avoid creating new silos. This article identifies four guiding factors for choosing tools: collaboration, learning, singleton tools, and Conway's law.

Over the past few years, I have worked with many organizations across different industry sectors to help them improve their effectiveness in building and operating software systems. Part of my role was to help these organizations to adopt practices like DevOps and Continuous Delivery. Choosing the right tools to support these practices is very important. Finding the right tool to use is not simply a case of comparing its advertised features or capabilities with those of a competitor—you need to consider the ways that a tool can affect multiple teams within an organization.

Rather than seeing a tool as purely achieving a particular task, you should look for ways in which a tool can be used to achieve higher-level aims, such as aligning goals between Development and Operations, improving engagement with business and commercial teams, and improving the quality of the software systems.

COLLABORATION

VERSION CONTROL

When considering the purpose of version control systems, you usually think of things like source code, change history, managing parallel work, audit logs, and so on. However, version control systems can be used in ways that help with communication and collaboration. Many organizations still have a separate team responsible for deploying and releasing changes, and often these teams are not very comfortable with version control.

However, using Github-style pull requests (or merge requests) along with a web-based front-end, developers and deployment support can collaborate on the changes, with a full history of discussions about the changes in the comments. In this scenario, Git feels less "scary" to the deployment support person, and they feel more empowered to say "yes" or "no" to a configuration change. They can sit down and explain to the developer why they think the config change is unwise, based on comments in the pull request and inline diffs of changes, all from a single web page.

When you emphasize change review via merge requests in a web UI, version control can help to foster collaboration and communication between teams, especially with more operations-focused teams that historically have less experience in version control.

DEPLOYMENT PIPELINE

The deployment pipeline is a central concept in Continuous Delivery, providing visualization and orchestration of all activities that take place between committing code to version control and changes appearing in production. A good deployment pipeline allows you to deploy both new features and bug fixes using the same process, and you can choose which kinds of testing to undertake before the changes go live. Deployment pipelines are typically responsible for things like software builds, running unit tests, deployments, test execution, and so on.

However, you can use the deployment pipeline for much more than these technical tasks. A well-chosen deployment pipeline tool allows us to model the existing workflow—including team roles—and show the progress visually. This means that a deployment pipeline can also be used to engage teams across the organization and reduce fear of change by visualizing current processes and showing their gradual evolution. This leads to increased collaboration and trust, making changes easier and more rapid, while providing an audit trace.

LOG AGGREGATION

Log aggregation has historically been seen as an operations concern, not one for developers. However, with the emergence of modern log aggregation tools (both on-premise and hosted), developers are realizing that logging has many benefits and is especially helpful for troubleshooting problems in production, where a debugger is not an option. Through the use of Event type IDs and explicit transaction tracing, developers can ditch the debugger and use logging instead, which means that developers and operations teams can use the same tool for diagnosing problems in any environment.

The use of log aggregation across all environments opens up collaboration between developers, testers, and operations

teams to enable much more rapid diagnosis and resolution of problems in production and upstream environments. It also creates more trust between teams because problems are found earlier.

LEARNING

The proliferation of new tools and techniques for DevOps and Continuous Delivery can be difficult to keep pace with. Many people involved in building and operating software systems are overwhelmed by the technologies and feel threatened when asked to use a new tool.

You must be careful not to leave people behind when choosing tools. Just as someone new to running might initially only run a few miles each day, increasing their distance over a period of months, you should choose tools in a way that encourages people and teams to "get started" with tools without insisting that they have to become an expert within a week. This sometimes means choosing tools that have *fewer features* than other tools, so that people can have time to learn, with the expectation that they may replace or augment that tool with a more advanced tool in 12 or 18 months' time.

This is particularly true for teams that have limited experience with automation. You need to allow teams to learn which tools and approaches work for them, rather than imposing some kind of "best of breed" tools that might be too difficult to use.

SINGLETON TOOLS

Tools that exist in only one environment can be particularly problematic when organizations try to adopt DevOps or Continuous Delivery. A "singleton tool" might be a database technology, a logging or monitoring platform, a load-balancer, or a virtualization technology that exists only in the production environment, usually due to the high cost of the licenses.

The problem with singleton tools is that they tend to create a divide between development and operations (aka "silos"). The special tool in production is seen by developers as "an Ops thing," and operations people see no reason why developers should have access to certain data because "it's an Ops thing." This breaks the vital feedback loop from production to development, limiting the improvements that can be made to the software.

Singleton tools make upstream testing much more difficult, meaning that high rates of failed deployments and buggy releases are effectively guaranteed.

CONWAY'S LAW

It's becoming increasing clear that the structure of teams within organizations, and the communication patterns between them, dictate the architecture of the software systems they build (Conway's Law). If you want a three-tier system, then have three teams (front-end, services, database). If you want microservices, let the teams have full-stack responsibility for one or more distinct business concepts, and so on. The impact of Conway's Law has often been focused on the application level, but less so at the level of the whole system, including operations and QA teams.

Some organizations—especially those with a single unified product or the ability to choose when all features go live—might want strong collaboration between development and operations teams so that a high change velocity is sustained. In this case, the development and operations teams would likely use many of the same tools—for instance: version control, ticketing, deployment tools, and configuration management.

Alternatively, let's say that another organization has an internal Infrastructure-as-a-Service (IaaS) team, providing a cloud platform running on Xen and OpenStack. In this case, you would want the development teams to interact with the IaaS team in a limited way, so that you preserve the "as-a-Service" relationship and don't blur the boundaries of responsibility. You would expect the use of shared tools here to be very limited and for other tools, such as monitoring, metrics, logging, and configuration, to be entirely separate between the two groups.

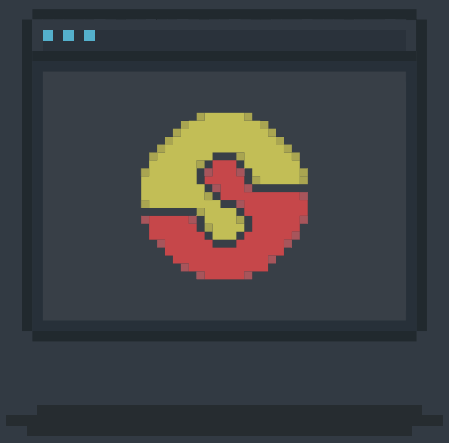
In some cases, it is better for teams not to share the same tool, particularly if a boundary of responsibility needs to be preserved. Collaboration through tools should match the boundaries of responsibility within and across teams, otherwise Conway's Law will start to take effect and modify the systems being built.

SUMMARY

You can choose tools for DevOps and Continuous Delivery in a way that significantly enhances the communication and collaboration between teams, even if a tool's main purpose is not collaboration. You may need to avoid tools that are too complex for teams at a given point in time, and you should prefer tools that can be present in all environments rather than "singleton tools" that exist only in production. You should also make sure that the use of shared tools aligns with the responsibility boundaries between teams in order to avoid confusion and the effect of Conway's Law.

MATTHEW SKELTON *has been building, deploying, and operating commercial software systems since 1998. Co-founder and Principal Consultant at Skelton Thatcher Consulting, he specializes in helping organizations to adopt and sustain good practices for building and operating software systems: Continuous Delivery, DevOps, aspects of ITIL, and software operability.*

MARTIAL ARTS
HAS BRUCE LEE.



AUTOMATED TESTING
HAS SAUCE LABS.

Maybe you can't do a one-fingered push-up, but you can master speed and scale with Sauce Labs. Optimized for the continuous integration and delivery workflows of today and tomorrow, our reliable, secure cloud enables you to run your builds in parallel, so you can get to market faster without sacrificing coverage.

Try it for free at saucelabs.com and see why these companies trust Sauce Labs.

Dropbox salesforce YAHOO! PayPal Bank of America VISA

sauce LABS

ELIMINATING ROADBLOCKS ON THE PATH TO CONTINUOUS DELIVERY

AUTOMATED TESTING PLAYS A KEY ROLE IN SUCCESSFULLY implementing Continuous Delivery. We've witnessed enterprises increasingly adopting fully automated delivery pipelines, successfully accelerating release cycles, achieving consistently high quality, and allowing their development teams to focus on writing software rather than on the mechanics of delivering it.

An example of an idealized, modern software delivery pipeline might look like the following:

- Plan user stories and manage issues with a project management tool like JIRA.
- Collaborate on code via GitHub pull requests or a code review tool.
- Kick off a build in a CI system like Jenkins or Bamboo.
- Automatically run unit and functional tests with open source testing tools like xUnit and other testing frameworks, and automation tools like Selenium and Appium.
- Deploy with an IT automation tool like Puppet or Chef, or using a PaaS.
- Monitor performance and impact on business metrics with systems like New Relic and Mixpanel.

Different organizations make different tool choices, of course,and there are usually a few pieces handled in a custom way due to the need to work with legacy systems or specialized processes. Whatever the challenge, software development teams have a thriving ecosystem of tools and services available to support a CD workflow. Indeed, it's this abundance of tool choices that is changing the equation and making Continuous Delivery possible for more and more teams.

Automated testing itself has come a long way as a part of this trend. Starting from early "test automation" tools designed to make QA teams more efficient, automated testing is now a critical part of automated delivery pipelines that are expected to run through complete test suites many times a day, with little tolerance for manual intervention, false failures, or infrastructural reliability problems.

Errors or bottlenecks introduced by automated testing infrastructure can break your build and block your deploy pipeline, creating expensive delays for software developers. Running automated tests rapidly and reliably is therefore critically important to a successful Continuous Delivery process.

By providing a high-reliability, scalable automated testing platform, we've been able to help enterprises sweep aside the time-consuming and error-prone maintenance of virtual machines and mobile devices, and allow them to instantly provision additional testing resources on demand. And we've prioritized fitting into the ecosystem of popular testing frameworks, CI systems, and surrounding tools and services, so that you can leverage existing investments and focus on optimizing your CD pipeline.

WRITTEN BY
Steve Hazel, *Cofounder, Chief Product Officer, Sauce Labs*

Automated Testing Platform by Sauce Labs

FUNCTIONAL AND UNIT TESTING

Sauce LABS

Sauce Labs provides a complete testing platform for native, hybrid, and web apps, allowing users to run Selenium, Appium, JS unit, and manual tests in any language on over 450 browser, OS, and platform combinations.

PRICING
By number of VMs and test run minutes

OPEN SOURCE
Yes

CASE STUDY HomeAway is a family of 50 websites and hundreds of applications that provide the largest collection of vacation rental listings in the world. One of the challenges they face is supporting the diverse set of devices and browsers on which people view their apps. They also feel pressure to speed up delivery time. HomeAway leverages Sauce Labs for production monitoring using an internal tool named Green Screen. Developers built Selenium scripts to execute primary customer user flows through Sauce Labs against their family of vacation sites. The objective is for these tests is to always be green; however, if a test fails, they receive an alert from Sauce Labs and the company responds. As a result, HomeAway has found the biggest value gained from using a combination of Sauce Labs infrastructure, real-user monitoring, and running their Selenium and Appium frameworks continuously so that quality isn't compromised.

CUSTOMERS

- Yahoo!
- Twitter
- Mozilla
- Salesforce
- Capital One
- Bank of America
- Zendesk
- Puppet Labs

CI TOOL SUPPORT

- Jenkins
- Bamboo
- Travis CI
- TeamCity
- CircleCI

TESTING SUPPORT

- Cross-browser testing
- Appium
- Mobile testing
- JavaScript testing
- Selenium
- Manual testing

LANGUAGE SUPPORT

C# Java JavaScript Node.js Perl Ruby PHP Python

BLOG sauceio.com | TWITTER @saucelabs | WEBSITE saucelabs.com

FOSTERING DEVOPS THROUGH A TESTING CENTER OF EXCELLENCE

BY JP MORGENTHAL

In case after case, businesses reveal that production issues and outages that occur when testing are limited due to time, when there are inconsistencies between the testing and production environments, and when too little attention is paid to areas outside of functionality, such as performance and security. When the business focuses on increasing testing and develops a Testing Center of Excellence, there is a 75% increase in the quality of deployments and less frequent production problems.

Given the name DevOps, it's natural for businesses to explore building their DevOps program by starting with a focus on development and/or operations. Ultimately, streamlining these areas is where the benefits of DevOps—faster time to market, greater quality in deployment, greater resilience in production—will be realized. Unfortunately, many of these programs are initiated under the guise of increasing speed and time-to-market, to the detriment of quality.

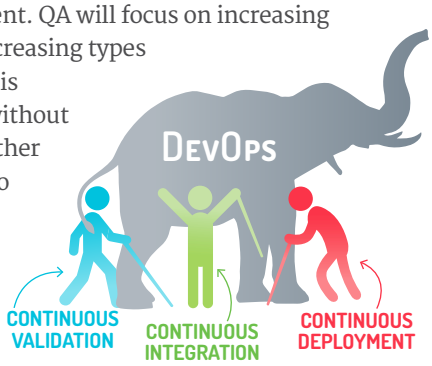
It is possible to streamline the ideation-to-operation lifecycle—moving from concept to production with as little friction as possible—in a way that increases time-to-market without needing to sacrifice quality. However, this cannot be accomplished through modest procedural improvements to individual areas of delivery. Instead, initiating improvement must start with a mission of quality and an analysis of risk.

The Testing Center of Excellence (CoE) is a team that fosters a best practices approach to forging this model of improvement, and it removes the need to sacrifice quality for speed.

FORGET WHAT DEVOPS IS, FOCUS ON DEVOPS GOALS

There is a lot of punditry surrounding DevOps, which often acts to confuse individuals looking to enact DevOps in their organizations. While it's all well and good that this media and analyst attention is building greater interest in DevOps, leading to more adoption, but the downside is the lack of definition and exploitive nature of vendors tends to lead to an inconsistent understanding among the various IT groups that need to be aligned in order to achieve the intended outcomes.

DevOps is truly the three blind men and an elephant scenario: each team is only interacting with one component, and is unable to identify the whole of their situation. Development will naturally focus on improving quality and speed of deliverables. Operations will focus on automation and repeatability of deployment. QA will focus on increasing throughput while also increasing types of testing. If one of these is approached in isolation without the others, it will force either a backlog or a wait state to occur. Agreement on the definition of DevOps is contextual to the role of the group. However, all groups can agree that each plays a role in transforming business requirements and ideas into production IT support as quickly and reliably as possible.



START WITH TESTING IN MIND

Quality assurance and testing are often deemed synonymous with regard to IT, but it is important to regard these as two distinct concepts. Quality assurance is about ensuring that the features and capabilities delivered align with the captured business requirements. This can be as simple as a checklist to say that a release contains all the features that were agreed to, or as complex as ensuring that each of those features operates as expected. For businesses that define quality assurance to include the latter set of responsibilities, testing will be the key tool to ensuring the implementation delivers what is expected.

By delineating these two concepts, it is easier to explain how testing is a shared responsibility. For example, only through appropriately designed unit tests can development minimize the number of defects in downstream testing. Moreover, quality assurance can focus on the usability and function of a particular release with testing while operations can focus on testing performance and security.

Surprisingly, very few organizations invest in managing this shared testing responsibility as a centralized practice, but instead delegate testing practices individually. The results are apparent in the billions of dollars lost every year due to downtime and outages. Hence, a key component of any DevOps strategy is the formation of a Testing Center of Excellence (CoE). The CoE has four areas of responsibility:

- **Practice Management** – The CoE manages the workflow for testing as it crosses group boundaries. They ensure that releases are not moved forward to the next stage until it meets established levels of acceptance criteria. They also work with the various teams to schedule resources, which is key in environments where there is resource contention for testing environments.

- **Governance** – The CoE is responsible for defining the testing goals and practices for any IT production deployment. It starts from the point of ideation, then moves to capturing and defining requirements, then to feature validation through quality assurance, user acceptance testing, and finally operational burn-in.
- **Organization** – The CoE defines the roles and responsibilities for those involved in testing. Each business has varying levels of experienced personnel and budget that can be applied to testing, so this group leverages matrix management techniques to define a testing organization.
- **Environment** – The CoE defines the configuration of the testing environment(s) for each project. Since one of the biggest issues plaguing continuous deployment and release management is configuration drift and inconsistent testing practices, the CoE works to ensure that testing is enacted on an environment that consistently moves closer to production the further along in the development lifecycle the project moves.

The CoE must be given the ability to halt the delivery process at any point if they believe quality is questionable. This will be a huge culture shift for many businesses, especially for businesses that are currently incentivizing the bad behaviors, such as on-time delivery over quality and defect management versus risk management. This is one of the key reasons that every DevOps program requires executive sponsorship in order to truly succeed.

TESTING IS NOT INTUITIVE

Formulating a Testing CoE can be a daunting task because of the lack of emphasis placed on testing science in software development. It will require finding a leader for the CoE that is passionate about this area of software engineering and that has the background and skills to set the standard and direction for the company with regard to testing. Even still, there are some potential pitfalls that can limit the value and/or success of the Testing CoE.

For a number of reasons that include a natural inclination to overlook common errors, testing is not intuitive. If ranked by importance, testing science barely makes the list. Testing is a very strategic goal and requires understanding of coding principles as well as production deployment considerations. Moreover, testing is a risk management initiative.

Establishing a testing plan for a release requires balancing time-to-market requirements and budget with the fallout from a failure in production.

> UNFORTUNATELY, MANY OF THESE PROGRAMS ARE INITIATED UNDER THE GUISE OF INCREASING SPEED AND TIME-TO-MARKET TO THE DETRIMENT OF QUALITY

A test plan is more than just "program X is supposed to Y based on first taking steps A, B, and C." It must validate that the essence of the business requirement is met. For example, can the user obtain the expected outcome if they use this feature, and what other outcomes are also possible? Are these outcomes desirable? Members of the Testing CoE must be able to formulate strategies to ensure this need is part of the testing processes.

Most developers understand the importance of unit testing their code, but few enjoy spending the time developing these unit tests. Hence, little time is given to actually thinking about trying to legitimately break their own code. This is the essence of unit testing design, and yet little training time is spent at the university-level preparing developers to deliver on this critical need. Members of the Testing CoE may be required to work individually with developers or provide additional group training to help developers better understand their responsibilities with regard to unit testing.

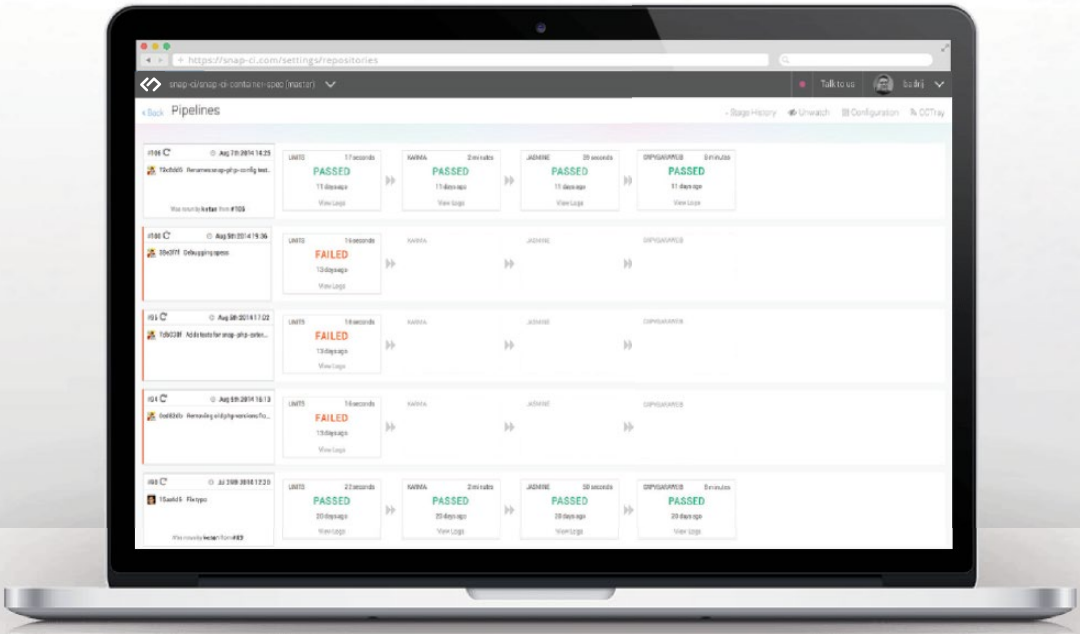
A TIPPING POINT

How far is your business willing to go to become agile and transform IT into the machine that will help you compete in an ever-changing world of technology? This question is being posed to business leaders on a daily basis. Executives have reached a tipping point with regard to their patience surrounding an inability of IT to deliver in a timely manner.

Through analysis of many businesses, we know which components are increasing IT operational overhead and hindering their ability to deliver: unplanned outages, break/fix, patching, skills shortages, and misaligned requirements. On the surface, the formation of a Testing Center of Excellence seems like an endeavor that will further delay releases and, in general, introduce more friction into the lifecycle. In the short-term, some of that is correct. However, the long-term benefits outweigh the inertia to get started, and ultimately lead to the reduction of many issues that are impacting Continuous Delivery. Moreover, the increased quality will be noticed by the business users, fostering continued confidence and increasing collaboration, which will have a direct impact on the overall competitiveness of the business.

JP MORGENTHAL *is an internationally renowned thought leader in the areas of IT transformation, modernization, and cloud computing. JP has served in executive roles within major software companies and startups. His expertise includes strategy, architecture, application development, infrastructure and operations, cloud computing, DevOps, and integration. He advises C-level executives on the best ways to use technology to derive business value.*

snap



Fail Fast: Evolve Faster

Automated, intuitive deployment pipelines set up in seconds

Deployment pipelines are a core part of the toolbox for teams doing continuous delivery, providing feedback at every step of the software delivery process. Snap's intuitive interface gets you up and running in seconds while retaining the ability to evolve as your team, product and processes change.

@snap_ci | https://snap-ci.com

Making production readiness of software visible every step of the way

Continuous Delivery is changing the way we develop software. While the bulk of the talk around this subject has been on the technical aspects of how to do it, the cultural changes around how various parts of an organizations need to collaborate are of equal if not greater importance. Thus it is natural that the tooling around Continuous Delivery attempts to ease this collaboration.

Deployment pipelines form the foundation of this tooling by modeling the complete path taken to release software to end users. They provide visibility into the production-readiness of every change made to your software. In addition, it also lets anyone self service any version of the software into an environment of their choice.

Setting up a deployment pipeline requires explicit tool support. Though Continuous Delivery as a practice builds Continuous Integration, most Continuous Integration tools lack the capabilities to build more than a visual equivalent of a deployment pipeline.

Besides, it is also common for teams to go "serverless": with both their version control systems and applications deployed in the cloud. To these teams, it makes sense that the deployment pipeline bridging their version control system to a production environment also lives in the cloud.

Fast, comprehensive feedback on the health of your software at every step from development through to production underpins Continuous Delivery.

Snap bridges this gap. It allows teams to set up Continuous Integration as the first steps of their deployment pipeline. Deterministic artifact propagation, fully customizable stages and build environments, manually triggered stages, stage execution parallelization and a carefully curated set of recipes offer powerful primitives to put together the exact deployment pipeline for each project, whether it be Ruby, Java, Scala, Clojure, Python, PHP, Javascript or a combination of the above.

WRITTEN BY
Badri Janakiraman, *Product Manager, ThoughtWorks Inc*

Snap CI by ThoughtWorks

CONTINUOUS INTEGRATION

snap

Snap CI is a continuous delivery and integration tool with automatic test parallelization and branch tracking builds and pull-request support. It also features first class support for deployment pipelines and simple in-browser build debugging.

Agentless	**Requires source control through GitHub**	**HOSTING** Cloud-hosted	**THIRD PARTY PLUGINS** Not supported

CASE STUDY Applauze.com is a San Francisco based startup that is known for their smooth pre-sales events for musical artists. Applauze needs to stay up during critical sales windows while also keeping their Rails based backend system up to date. Recently their team upgraded this system from Rails 3 to Rails 4 with the upgrade work done on a branch. Snap's automatic branch tracking kept the team informed about the health of this work. More importantly, Snap's Integration Pipelines ensured that the upgrade branch would merge back into the master with passing tests. This gave Applauze the confidence required to undertake this work while staying live. Applauze rarely use long lived branches for their day to day work. However, in this case, Snap allowed the team to manage the risk of doing a significant upgrade without the attendant risk of an impossible merge.

SUPPORTED PLATFORMS
- Java Runtime Environment
- Ruby Environment
- PHP Environment
- Python Environment
- Javascript/V8 environment

CM INTEGRATIONS
None

CUSTOMERS
- BBC
- GNS Science, New Zealand
- Escape the City
- Applauze
- PlayOn! Sports
- Envato
- Lifeletters
- RepairShopr

FEATURES
- Sets new builds to be the trigger for new deployments
- Automatic validation builds for code changes to the master branch
- Ability to change the default configuration/template for the standard build machines

BLOG blog.snap-ci.com | TWITTER @snap_ci | WEBSITE snap-ci.com

Are Containers Part of Your IT Strategy?

BY JIM BUDGWADIA

Containerization allows application components to be portable to any cloud provider that offers a base operating system that can run the container. Using containers avoids deep lock-in to a particular cloud provider, or a platform solution, and enables application runtime portability across public and private clouds. There were some key points in the last decade that lead to wider adoption of this technology: mainstream server virtualization and efficient image management for Linux containers.

In 2002, VMware introduced their Type 1 hypervisor which made server virtualization mainstream and eventually a requirement for all enterprise IT organizations. Although cost savings are often cited as a driver, virtualization became a big deal for businesses as it allows continuous IT services. Using virtualization, IT departments could then offer zero-downtime services at scale and on commodity hardware. In 2014, Docker, Inc. released Docker 1.0. Docker provides efficient image management for Linux containers, and provides a standard interface that can be used to solve several problems with application delivery and management. Much like VMware made virtualization mainstream, Docker is rapidly making containerization mainstream. In this article, I will discuss four reasons why you should consider making containerization part of your business strategy.

CONTINUOUS DELIVERY OF SOFTWARE

Virtualization enabled the automation and standardization of infrastructure services. Containerization enables the automation and standardization of application delivery and management services (a.k.a. platform services). Faster software delivery leads to faster innovation. If your business delivers software applications as part of its product offerings, the speed at which your teams can deliver new software features and bug fixes provides key competitive differentiation.

Virtualization, service catalogs, and automation tools can provide self-service and on-demand virtual machines, networks, and storage. However, rapid access to virtual machines and infrastructure is not sufficient to deliver applications. A lot of additional tooling is required to deliver applications in a consistent and infrastructure agnostic manner.

Application platform and configuration management solutions have tried to address this area, but have not succeeded on a large scale, as until recently there was no standard way to define application components. Docker addresses this gap, and provides a common and open building block for application automation and orchestration. This fundamentally changes how enterprises can build and deliver platform services.

Another fast growing trend is that cloud applications are being written using a microservices architectural style, where applications are composed of multiple co-operating fine-grained services. Containers are the perfect delivery vehicle for microservices. Using this approach, your software teams can now independently version, test, and upgrade individual services. This avoids large integration and test cycles, as the focus is on making incremental, but frequent, changes to the system.

APPLICATION PORTABILITY

Businesses are adopting cloud computing for infrastructure services. Public cloud providers are continuously expanding their offerings and are also constantly reducing their pricing. Some cloud providers may have better regional presence, and others may offer specialized services for certain application types. At a certain spending point, and for some application types, private cloud remains an attractive option. For all of these reasons, it makes sense to avoid being locked into a single cloud provider.

DevOps CULTURE

The DevOps movement builds on Agile software development, where small incremental releases are favored to long release cycles, and the Lean Enterprise philosophy, where constant customer feedback loops are used to foster a culture of innovation.

With DevOps, developer teams can also be responsible for what are seen as traditional operations roles. As Adrian Cockroft explains, the traditional definition of when a project is "done" was when the code was

> "Docker provides a common and open building block for application automation and orchestration."

released to production. Now, "done" is when the code is retired from production.

However, implementing DevOps practices for a startup delivering a single web application will be very different than DevOps implementation for an enterprise delivering several applications. In larger environments, and for more complex applications, a common platform team is required to service multiple DevOps teams.

Containerization provides a great separation of control across DevOps and platform concerns. A container image becomes the unit of delivery and versioning. DevOps teams can focus on building and delivering containers, and the platform team builds automation around operating the containerized applications across public and private clouds, as well as shared services used by multiple DevOps teams.

COST SAVINGS

Virtualization allows several virtual machines to run on large physical servers, which can lead to significant consolidation and cost savings. Similarly, containerization allows several application services to run on a single virtual or physical machine, or on a large pool of virtual or physical machines.

Container orchestration solutions can provide policies to packing different types of services. This is exactly what Platform-as-a-Service (PaaS) vendors, like Heroku, have been doing under the covers. Containerization orchestration tools that are built on open technologies like Docker can now make this transparent to end users, and also pass along the cost savings to their customers.

CURRENT CHALLENGES

Recently, perhaps influenced by the buzz around Docker, Google revealed that all of their applications, from Search to Gmail, run in Linux containers. However, Google and others have spent several years building and fine-tuning platforms and tools around containers, and until recently they have treated these tools as a competitive advantage.

For mainstream adoption of containerization, better general purpose container orchestration and management tools are required. Application networking and security also remain areas of key development. Finally, the options for non-Linux applications are currently limited. These are the kinds of challenges we expect to see solutions for in the coming year and beyond, whereas this market is currently dominated by a select few, with Docker at the head.

SUMMARY

The adoption of infrastructure virtualization has greatly contributed to and enabled continuous IT services, including the spread of Continuous Delivery and DevOps best practices. Containerization drills down into these DevOps practices and helps to enable continuous application delivery. Containerization also provides application portability, and can be a key architectural building block for cloud native applications. Once an application is containerized, the containers can be run on a pool of virtual or physical machines, or on infrastructure-as-a-service (IaaS) based public clouds.

> "While no single technology will be a silver bullet, containerization can help with streamlining your application delivery pipeline."

For new applications, packaging the application components as containers should be strongly considered. Just as with virtualization, the list of reasons why not to containerize are already rapidly shrinking. Another use for containerization is to transform traditional applications that now need to be delivered as Software-as-a-Service.

While no single technology will be a silver bullet, containerization can help with streamlining your application delivery pipeline. If your business delivers software, you can leverage containerization to develop and operate software more efficiently and in a highly automated fashion across public and private clouds.

JIM BUDGWADIA *is one of the founders of the cloud-based startup Nirmata. Jim was previously part of the Cisco team that helped make the company the foremost cloud consulting company in the world. As a developer, he enjoys building and delivering solutions for cloud, virtualization, data center networking, and wireless systems.*

Ready for CAMS: The Foundation of DevOps

Recent studies show that organizations adopting DevOps practices have a significant competitive advantage. They react quicker to changing market demands, get out new features faster, and are better at executing changes. Though the definition varies, we think about DevOps in terms of CAMS: **Culture**, **Automation**, **Measurements**, and **Sharing**. In order to focus the entire team on performance, you *must* plug performance into those 4 pillars.

Culture - Tighten the Feedback Loops Between Development and Operations

Culture is difficult to change, but extremely important, as it's the way teams *work together* and *share the responsibility* for end-users. It encourages the adoption of agile practices in operations, allows developers to learn from real-world Ops experiences, and breaks down the walls between teams. Dynatrace aids collaboration by providing a shared language between Ops, Test, and Dev. This means clearly stated performance requirements across teams, which eliminates finger pointing to the root cause of performance issues.

Automation - Establish a Practice of Automated Performance Testing in CI

Ops and Test teams have a good understanding of performance. They need to educate developers on the importance of performance in large-scale environments under heavy load, making devs aware of recent performance problems and solutions. This makes common problem patterns easier to prevent. Dynatrace helps identify those patterns in production environments as well as in development stages to prevent them from making it into production.

Measurement - Measure Key Performance Metrics in CI, Test and Ops

With performance aspects covered in earlier testing stages, performance engineers can focus on large-scale load tests in a production-like environment. This helps find data-driven, scalable, and 3rd party impacted performance problems. Close collaboration with Ops ensures that tests can be executed in the production environment or a staged environment that mirrors production. Executing these tests in collaboration with Ops means more confidence when releasing a new version. From an Ops perspective, defining a set of key performance metrics, monitored in all stages and agreed on by all, allows for better collaboration in the future.

Culture is the way teams work together and share the responsibility for end-users.

Sharing - Share the Same Tools and Performance Metrics Data Across Dev, Test, and Ops

Traditional testing teams are familiar with executing performance and scalability tests in their own environments at the end of a milestone. With less and less time for testing, test frameworks and environments have to be available to other teams to make performance tests part of an automated testing practice in a Continuous Integration environment. Automatic collection and analysis of performance metrics, as done by Dynatrace, ensures that all performance aspects are covered. This once again entails defining a set of performance metrics applied across all phases, as this helps identify the root cause of performance issues in production, testing, and development environments.

by Wolfgang Gottesheim,
Technology Strategist, Dynatrace

dynatrace

DIVING DEEPER INTO

CONTINUOUS DELIVERY

TOP TEN #DEVOPS TWITTER FEEDS

 @jezhumble

 @RealGeneKim

 @MrHinkle

 @martinfowler

 @andrewsmhay

 @lusis

 @adrianco

 @allspaw

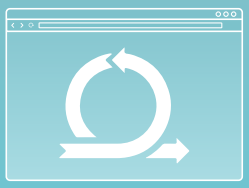
 @garethr

 @davefarley77

DZONE CONTINUOUS DELIVERY ZONES

DevOps Zone

DevOps is a cultural movement, supported by exciting new tools, that is aimed at encouraging close cooperation within cross-disciplinary teams of developers and IT operations/system admins. The DevOps Zone is your hot spot for news and resources about Continuous Delivery, Puppet, Chef, Jenkins, and much more.



Agile Zone

In the software development world, Agile methodology has overthrown older styles of workflow in almost every sector. Although there are a wide variety of interpretations and specific techniques, the core principles of the Agile Manifesto can help any organization in any industry to improve their productivity and overall success. Agile Zone is your essential hub for Scrum, XP, Kanban, Lean Startup and more.

Cloud Zone

The Cloud Zone covers the host of providers and utilities that make cloud computing possible and push the limits (and savings) with which we can deploy, store, and host applications in a flexible, elastic manner. This Zone focuses on PaaS, infrastructures, security, scalability, and hosting servers.

DZONE CONTINUOUS DELIVERY REFCARDZ

Preparing for Continuous Delivery bit.ly/1EbzQjm

This Refcard is written to ease that transition, giving guidance, advice, and best practices to development and operations teams looking to move from traditional release cycles towards Continuous Delivery. Also includes a comprehensive checklist to ensure that your transition will proceed smoothly.

Git Patterns and Antipatterns bit.ly/1EbAldh

The most popular Distributed Version Control System can also be the most unwieldy. Migrating to Git requires a smart transition of team development practices without degrading code quality or introducing security risks.

Deployment Automation Patterns bit.ly/1yTuwN6

This card covers 7 patterns, and includes useful code snippets as well as antipatterns.

Continuous Delivery: Patterns and Antipatterns in the Software Lifecycle bit.ly/1B0CCBH

Continuous Delivery is the topic covered in this DZone Refcard, and is loaded with over 40 different patterns and antipatterns for configuration management, continuous integration (CI), infrastructure, incremental development, and deployment. Continuous Delivery: Patterns and Antipatterns in the Software Lifecycle encourages you to follow the mantra: Release early and often.

TOP 3 CONTINUOUS DELIVERY & DEVOPS WEBSITES

continuousdelivery.com

devops.com

dev2ops.org

DZone PRESENTS

CONTINUOUS DELIVERY visualized

Continuous Delivery advocates the creation of maximally automated deployment pipelines.

This visualization of an optimally (but not entirely) automated deployment pipeline shows how Continuous Delivery works.

Each stage (big circle) is composed of multiple activities (little circles). Each activity can be automated or otherwise facilitated by various (mostly open-source) tools. We surveyed our audience to see which tools they used for which deployment-related activities. The three most commonly used tools are listed next to each activity.

- Any long-running step, such as UAT, Pre-Production testing, or Exploratory Testing, can happen even after the change has already been deployed to Production.
- If significant issues are found in any long-running step, and the change has not been deployed to Production, the team should manually halt the pipeline.
- If significant issues are found in any long-running step, and the change has already been deployed to Production, the team should rollback Production to the last working release.

Diagrams are based on Jez Humble's diagrams from the Continuous Delivery blog (http://continuousdelivery.com/2010/09/deployment-pipeline-anti-patterns)

Special thanks to Matthew Skelton for helping build these diagrams.

REFERENCE KEY

automated trigger — manual trigger — optional step

automated execution — manual execution

1 2 3 three most commonly used tools used for this activity

START

01. commit
- syntax check: 1 Checkstyle, 2 Eclipse, 3 SonarQube
- unit tests: 1 JUnit, 2 TestNG, 3 NUnit
- auto-merge for developer branches (when using Git Flow, etc.)
- compile (for compiled languages)
- code metrics: 1 SonarQube, 2 Cobertura, 3 FindBugs

02. automated acceptance testing
- integration tests: 1 JUnit, 2 TestNG, 3 Jenkins
- story-level tests: 1 Cucumber, 2 JUnit, 3 Selenium
- feature-level tests: 1 Selenium, 2 JUnit, 3 Cucumber
- BDD framework tests: 1 Cucumber, 2 JBehave, 3 SpecFlow
- component tests: 1 Mockito, 2 TestNG, 3 NUnit
- stubbed and mocked endpoints of data: 1 Mockito, 2 EasyMock, 3 SoapUI

03. exploratory testing
- visual (UI) tests: 1 Selenium, 2 manual, 3 Visual Studio
- usability tests: 1 Selenium, 2 SoapUI, 3 Cucumber

04. user acceptance testing (UAT)
- showcases
- feature-level testing by the client: 1 manual, 2 Selenium, 3 Cucumber

05. staging and pre-production
- network tests: 1 JMeter, 2 Wireshark, 3 Ping
- post-deployment tests: 1 manual, 2 Selenium, 3 SoapUI
- capacity tests: 1 JMeter, 2 LoadRunner, 3 Gatling
- performance tests: 1 JMeter, 2 LoadRunner, 3 Gatling
- smoke tests: 1 Selenium, 2 JUnit, 3 Cucumber

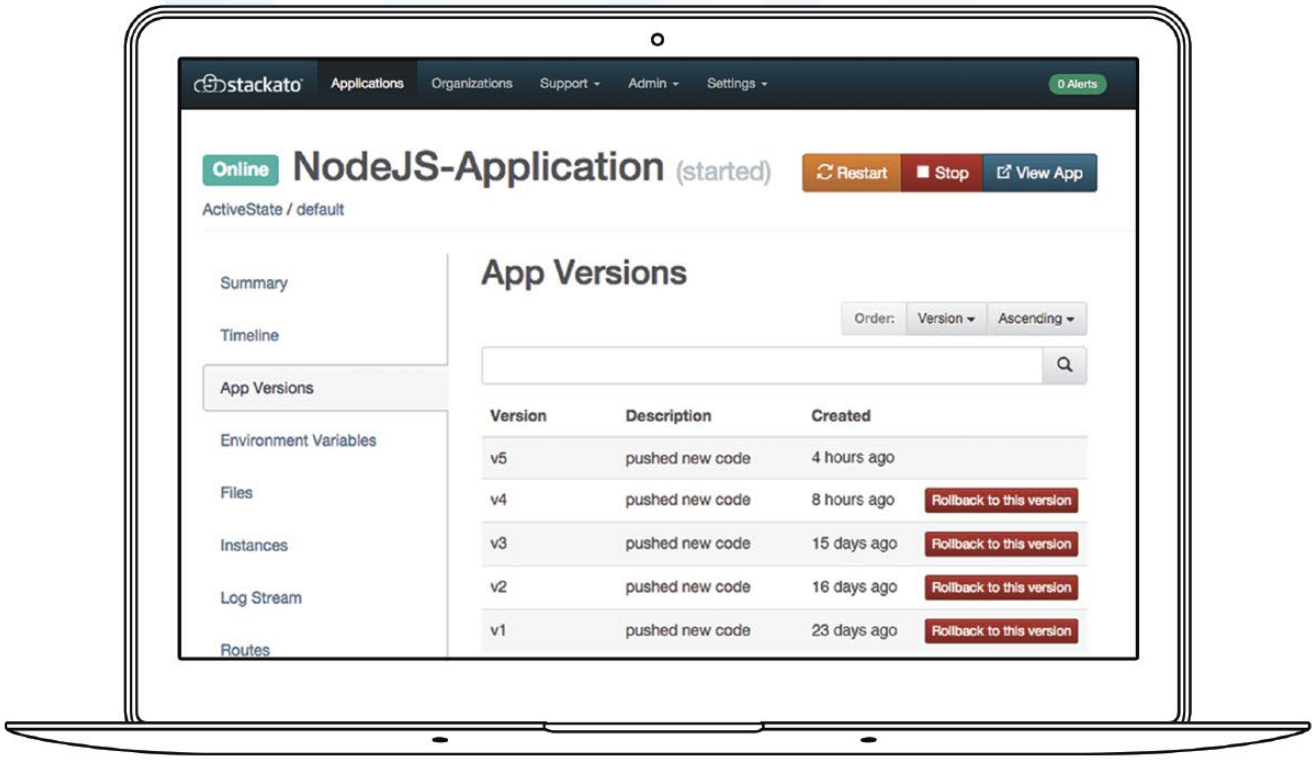
06. production
- rollback and redeploy: 1 Jenkins, 2 Bamboo, 3 Chef
- post-deployment tests: 1 manual, 2 Selenium, 3 SoapUI
- smoke tests: 1 Selenium, 2 JUnit, 3 Chef
- ongoing live transaction tests
- you did it!

DZone
from dzone.com

stackato®

by ActiveState®

The Platform for the Agile Enterprise



Achieving Continuous Delivery with PaaS

Continuous Delivery requires an automated way to deploy and manage applications. Stackato integrates with your CI tools to automate software provisioning, scaling, logging and monitoring, while maintaining consistent environments across the development lifecycle.

Built on Cloud Foundry, Docker and other leading open source components, Stackato lets you release more rapidly and innovate faster.

Get started FREE up to 20GB at stackato.com

Implementing Platform-as-a-Service (PaaS) in a Continuous Delivery Workflow

As a colleague of mine once said, not everyone can "pull a Netflix." They have set a high bar in a number of areas of software development, one of which is automating the application delivery process, by creating their own tools to help achieve Continuous Delivery.

While not every company has the organizational structure to build their own tools, many companies still want to implement Continuous Delivery. Delivering production builds as frequently as possible increases your responsiveness to customers' demands, and ultimately gives your company a competitive edge in the market.

Continuous Delivery requires somewhere to deliver the software to. That's where PaaS (Platform-as-a-Service) comes in: organizations can automate deployment of their applications to a platform that provisions services and automates instance scaling, logging, monitoring, and other pieces of the application lifecycle. As a PaaS software provider, we have found a lot of interest in using Stackato in conjunction with Continuous Integration tools such as Jenkins and Bamboo.

A PaaS will typically have an API interface which is accessible through a CLI client or client libraries. A step can be added to the end of a CI software build to push the software somewhere it can be inspected in a running state. You can do this by scripting the CLI client, or by using a plugin. The Cloud Foundry plugins for Bamboo and Jenkins make it possible to deploy a tangible, testable, working copy of web applications every time a build is triggered.

"YOU MAY ALREADY BE USING CI SYSTEMS FOR AUTOMATED TESTING...CONNECTING THESE SYSTEMS TO A PAAS CAN DELIVER NETFLIX-LIKE VELOCITY."

You may already be using CI systems for automated testing in development and quality assurance. Connecting these systems to a PaaS can deliver Netflix-like velocity without the significant investment in custom automation tooling.

[Learn more about the Cloud Foundry Jenkins Plugin.]

WRITTEN BY
Troy Topnik, *Technical Communications Manager*

Stackato by ActiveState

CONFIGURATION MANAGEMENT

Stackato is a commercially supported PaaS that leverages open source components such as Cloud Foundry and Docker, and integrates with Jenkins, Bamboo and other CI systems. It runs on top of any cloud as a self-service platform for deploying, updating and managing applications.

Agentless model

PROTOCOLS
- SSH
- HTTP
- HTTPS

HOSTING
Cloud-hosted or on-premise

CASE STUDY Service-Flow develops and provides Software-as-a-Service (SaaS) for IT service integration, relied on by financial and insurance companies serving millions of customers. To enable rapid development of Java applications without having to manage their own servers, Service-Flow decided to implement Platform-as-a-Service (PaaS) middleware. After using a hosted PaaS solution, Service-Flow moved to Stackato, a self-service private PaaS that supports deployment on Amazon Web Services (AWS). Using Stackato, Service-Flow runs 60+ mission-critical applications, achieves zero downtime, supports continuous delivery with integrated Jenkins, and ensures security of their customers' data on AWS private cloud.

CUSTOMERS
- ExactTarget
- Mozilla
- MTN Communications
- Service-Flow
- HP
- Cisco
- Nelnet
- Angie's List

FEATURES
- Automatically provisions databases and services
- Can deploy apps directly from CI system, IDE or CLI client
- Easy management of app instances and environments

DATA STORES
- MySQL
- PostgreSQL
- MongoDB
- Redis
- Memcached
- OracleDB

BLOG activestate.com/blog | TWITTER @activestate | WEBSITE activestate.com/stackato

Beyond Tooling and Process: the Culture of Continuous Delivery

BY ANDREW PHILLIPS

Today's articles and conversations about Continuous Delivery tend to revolve around the technical aspects: "Which is better: Puppet, Chef, or Salt? Should I use Jenkins, Go, or XL Release? How do I build a CD pipeline with containers?" and so on. If we're looking at CD properly, though, this is the sideshow—an implementation detail at best. The real Continuous Delivery story is much bigger and has the potential to completely transform your business.

Creating a Modern Consumer Experience

Don't get me wrong: as a technical kind of person, I get very enthusiastic about the tech. There are a lot of cool tools and frameworks out there, and the tooling landscape is expanding rapidly. Working with evolving technology is a lot of fun, and there are plenty of challenges to be solved around supporting Continuous Delivery at scale.

However, the fact that we now have a set of tools to automatically build and deploy applications, provision environments, run tests, and more, is not what I think is important about CD. Even the notion of wiring all these tools up efficiently to create delivery pipelines isn't all that interesting. To me, the thing that is really exciting about Continuous Delivery is that it can allow us to fundamentally change the way we interact with our users. I think CD can finally allow us to turn software delivery into something approaching a modern consumer experience.

In short, Continuous Delivery isn't a toolset, and it isn't a business process either. Continuous Delivery is a *new way of doing business*. What does this "new way of doing business" look like? What is the *mindset* or the *culture* of a Continuous Delivery organization? To me, these are the most important aspects: **focus on the end user** and **improve through data**.

Focusing on the End User

Having a focus on the end user sounds like an obvious statement that every organization should claim to aspire to, but given the way we release software today, there is often an enormous gap between the people creating the software and those that actually use it. I've worked on numerous teams where we never really knew who was actually going to use the software, or why. We certainly didn't have any personal contact with the end users of our software, and thus had little ability to empathize with their needs and constraints, or put ourselves into their shoes to try to understand how the system could work best for them.

> "Continuous Delivery isn't a toolset. Continuous Delivery is a *new way of doing business.*"

This gap, which was reinforced by organizational structures that put lots of layers between developers and users, resulted in something like an "emotional variant" of Conway's Law: not only did we build systems that reflected the layers of organization, we developed mentalities and boundaries of empathy to match: "The system is totally unworkable for the users? Ok, but look how clean our repository layout is and how elegantly we've managed to decouple these components and how complete our test coverage is…"

If your team knows your users, if you have some kind of personal connection with them, meet them from time to time to talk about software, but also about baseball, family, your next vacation, or whatever… then you cannot feel proud of a system that is totally unworkable for them.

But it's not enough to just create an emotional connection. In order to turn the intrinsic motivation into great software and a great user experience, every member of the team needs to understand how the overall service they are creating is put together and delivered to the user. They also need to be able to contribute to improving the service if they see an opportunity to do so—whether it's part of their area of expertise or not.

If you're thinking that this sounds suspiciously like "product teams" or "end-to-end teams" or "DevOps,"

then you're on the right track. The key point here is that this goes beyond development, QA, and operations to include the business and, beyond them, the end users for whom the software is actually being built. The aim of giving everyone visibility over the entire process, and the ability to influence it, is not because more pairs of eyes can spot more opportunities for optimization; it's because a more engaged team with a greater influence on the delivered service and a stronger connection to the end user is more motivated to look for improvements to begin with.

If we have a team that is continuously motivated to make things better for our end users, how do we identify where the opportunities for improvement are in the first place? If we have given them the chance to make changes, how do we figure out whether those changes are helping? In a word: data, data, and more data.

Improving Through Data

It's not as though we don't have any data today, of course. Most organizations collect tons of information on many aspects of system behavior. The vast majority of this information is of a technical, operational nature. In most enterprises, it's much easier to figure out whether a particular CPU in your datacenter is doing okay than it is to understand whether a particular user of your services is satisfied.

> "How do we identify where the opportunities for improvement are in the first place?"

Which of the two pieces of knowledge is more important for your business? And if it's the user that really matters, then why are we basing so many of our decisions about the user-facing functionality and behavior of our services on educated guesses and gut feelings, without any kind of follow-up plan to figure out whether we guessed right?

Improving through data means applying the same methodology used for improving the technical performance of our systems to making services better for our users:

1. Measure to identify the pain point
2. Formulate a hypothesis as to what is causing the pain point
3. Make a small change to a single part of the system based on the hypothesis
4. Measure to check for the expected improvement
5. Rinse and repeat

So: it's the all-night feature brainstorm sessions, but with accurate, detailed information on how your users are actually interacting with your services. This means feature requests with measurable estimates of how the feature is supposed to affect users, and service architectures that allow small changes to be quickly and efficiently deployed to production, and reverted if necessary. This means applications must be able to collect data that can be analyzed to determine whether the change had a positive effect on user experience.

Everyone Else is Doing It

If you're thinking that this all sounds very nice but will certainly not happen anytime soon in your organization, bear this in mind: you're probably working with people in your own organization *that are doing this today*. Tell this story to your colleagues in Marketing or even Human Resources and you're quite likely to hear that this is all rather old hat.

Marketing designs advertising campaigns by analyzing the effect of previous efforts, employing A/B testing, and using full instrumentation to track every user interaction and collect all relevant metrics. Human Resources have personal development programs tailored to individual preferences and learning styles, and training courses that adapt to your current performance and suggest additional programs that might be useful for you. The resources for exploring user experience and satisfaction are all pretty well-established in other departments.

The big realization here is that all these resources are powered by software! *The stuff that we are writing*. It's more than a little ironic that we have made it possible for so many fields to revolutionize their ways of working, yet have so far been unable to provide a better user experience for ourselves. That, to me, is what Continuous Delivery is all about: revolutionizing the way we serve our users, and making the delivery of IT services a modern experience.

ANDREW PHILLIPS *is VP of Product Management with XebiaLabs, and a cloud, Continuous Delivery and automation expert. When not "developing" in PowerPoint, he contributes to open-source projects including Apache jclouds, a leading Java cloud library, and co-maintains the Scala Puzzlers website.*

The real disruptors? Your customers.

Welcome to the application economy. Where customers don't walk into a store. They pull out their mobile phones. Where users don't just expect a flawless experience, but one tailored to their tastes. Where the difference between success and failure come down to quality of your applications—and billions in potential revenue hangs in the balance. It's an enormous opportunity for those nimble enough to seize it. And CA Technologies helps you do just that. From planning to development to management to security, only CA offers an end-to-end portfolio of software solutions to give your business a competitive advantage in the application economy.

See how leading businesses are driving double-digit growth.
Read CA's new Application Economy Market Study at **rewrite.ca.com/appecon**

ca technologies

Business, rewritten by software™

Continuous Delivery – The Ultimate Competitive Differentiator

Today, every business is a software business. As the saying goes, the competition is only a click away. Today's most successful companies are now laser-focused on finding ways to deliver new applications, services, and capabilities faster and at a higher quality than ever before.

In order to achieve this acceleration, many IT groups are embracing the concepts of DevOps and Continuous Delivery, which offer application delivery methods that speed up delivery while improving communication, integration, and collaboration between development and operations. This doesn't happen overnight. It's a journey that involves the transformation of organizations, people, processes, and technology.

Many of our enterprise customers started on this journey by automating their manual release processes with custom written scripts, which are inflexible, error prone, and a challenge to maintain. They soon realized that something more was needed.

An enterprise-class release automation solution is a significant accelerator at the start of any DevOps journey. It should automate the deployment of applications across your entire software delivery pipeline and provide features that make it easy to ensure the use of best practices. Release workflows should be built from reusable components to ensure consistency and widespread adoption.

An enterprise-class release automation solution is a significant accelerator at the start of any DevOps journey.

Our customers are learning that a Continuous Delivery solution built for the enterprise can be the ultimate competitive differentiator. It enables their development teams to experiment, receive rapid feedback, and be more innovative. This translates into higher quality software, and as such, represents a massive opportunity for companies to distance themselves from their competitors.

WRITTEN BY
David Cramer, *VP, Product Management Application Delivery, CA Technologies*

CA Release Automation by CA Technologies

RELEASE AUTOMATION

CA Release Automation simplifies management of complex releases, ensures continuity of release execution processes and mitigates risk of failure by orchestrating and automating multi-tier workflows across users, applications and environments.

THIRD PARTY PLUGINS Does support 3rd-party plugins

HOSTING On-premise

CI INTEGRATIONS
- Jenkins/Hudson
- JetBrains TeamCity
- Microsoft Team Foundation Server

MODELS Push or pull agent model

SUPPORTED PLATFORMS Java Runtime Environment

CASE STUDY A major healthcare organization decreased deployment times from hours to minutes and reduced the amount of time spent troubleshooting quality issues. Together this allowed the team to significantly reduce time to market for new releases of its solution. By standardizing and streamlining release processes, deployment resources can now facilitate multiple application deployments within a limited deployment window as opposed to multiple resources working several hours to achieve the same results.

FEATURES
- Pipeline view of entire delivery process
- Ability to compare environments and releases
- Multiple builds rolled into single deployment package to handle micro services architecture
- Integrations to over 120 open source and enterprise tools

CUSTOMERS
- City Index
- CSC
- HCL Technologies
- Liquid Net, Inc.
- Logicalis SMC B.V.
- Molina Healthcare, Inc.
- SwissCom

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

BLOG blogs.ca.com | TWITTER @cainc | WEBSITE ca.com

CONTINUOUS DELIVERY: From Theory to Practice

BY YARON PARASOL

This article contains two more diagrams that we are not able to show in this guide. You can view the full article with its original diagrams in the DevOps Zone.

When we discuss Continuous Delivery (CD), the starting point would need to be the motivation that has driven this IT evolution. The demand for Continuous Delivery was brought on by businesses' need for more agility and faster time to market, where time to market is the primary motivation, and agility is the means to achieving that. Another two important driving factors are the need for more efficient feedback loops, as well as the reduction of work in progress/process (WIP). WIP is a term from the inventory world, which is a concept that refers to a company's partially finished goods waiting for completion and eventual sale; goods that are unable to reach the market can eat up company resources such as storage and bound capital.

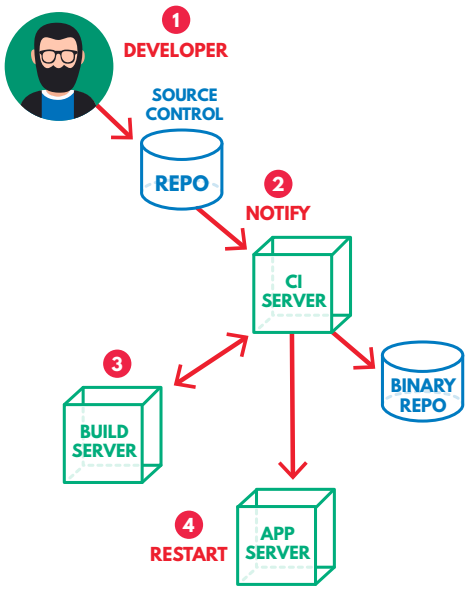
This is why the whole Agile methodology, with sprints and Scrum, was even started in the first place. This led to the development of continuous integration (CI) to automate build processes, speed up delivery, minimize human involvement, and prevent error. However, even with CI, most deployments still remained manual. At best, parts of the deployment process were automated, but only in the context of small closed environments (basically non-configurable environments).

However, if the ultimate goal is to get new features to production in a matter of hours (rather than months or half a year), then there is still a need to fill the gap between Agile development and CI, and actually pushing frequent, small updates to production. Getting the code to production is where Continuous Delivery comes to complete the entire cycle from development to market. You want to be able to push a minimal scope of new code in a way that is easily testable and with limited exposure, but also be able to monitor it closely, and roll it back if it's not good.

CONTINUOUS INTEGRATION IN THE REAL WORLD

While the motivation and methodology is generally clear, the actual execution is another story entirely. With CI/CD we're talking about a streamlined process that's quite complex. This means we don't want human intervention, or at the very least to minimize it significantly, and we want to extend the automation all the way to production, all while combining this with testing and intelligent monitoring.

Let's take a look at CI what a normal CI process looks like.



In this diagram, the developer (1) is first pushing the code to a repository for source control, for which the CI server is listening (2). The CI server then invokes the build process (3), where the result is a binary or binary repo. The binaries are uploaded to the application server and either it is then restarted, or it is removed from the load balancer and the process is then reloaded as needed (4). This is of course a simplified explanation for now. It's also important to note that there's unit and integration testing happening between steps 3 and 4.

IAAS AND CONTAINERS IN THE CI PROCESS

After you've built a working automated build process, there is another important challenge to overcome before being able to push to production, and this is the matter of controlled environments. This is basically the primary reason Infrastructure-as-a-Service (IaaS) and containers have become very popular in the CI world. The three leading challenges with controlling environments have been working in clean environments, ensuring proper utilization of resources, and parallelization of processes. This is due to a number of reasons:

- "Dirty" machines (previously install packages and unclean configuration) are bound to cause inconsistencies with running tests and building binaries.
- To ensure environments aren't manually set up and occupied unnecessarily for extended periods of time, resulting in others not being able to use them.
- To enable parallelization of the process by spawning a set of VMs/containers and the running of different test suites/builds on each.

For these reasons, with typical integration and testing, it has become popular to use IaaS. With IaaS, you can build the on demand resources you need (compute, storage, and network resources) within just a few minutes; if something goes wrong, you can easily wipe the entire environment, and do it all over again.

Of course, the most ideal way is to automate these processes. You can choose the DIY method of using scripts, and install the application stack using configuration management tools, or even more scripts. This means starting the application components, cloning a git repo, and running the build process, which then yields a binary or package with the updated code. Then you have your binary (which is really only half the process). Then you need to install the binaries on the environment you just created and run the integration tests. The benefit is that it's a robust process that is less error prone, it solves the issue of clean environments, and it enables parallelization of on-demand resources.

The disadvantage of using IaaS is again the under-utilization of resources—and that's where containers come into the mix. Using containers is similar to VMs, but more lightweight; if a VM takes minutes to load, a container can be spawned pretty much instantly. It's a fresh environment, utilizes minimal resources, and enables parallelization—and since it's so lightweight, the binaries are also released much more quickly.

CONTINUOUS DELIVERY... FROM THEORY TO PRACTICE

After we've taken a look at typical CI, and then focused on CI in an IaaS environment or with a containerization flavor, we can segue into CD.

You might assume that after you have the application packaged and tested, it should be pretty easy to deploy to production, right? Well the answer is actually no. Unfortunately, this has proven quite difficult in real production environments.

There are several obstacles when it comes to zero time deployments.

After you've automated your build process, it's not just a matter of automatically deploying your code to production. For starters, the build process is not done in production, so ultimately it has no business risk. When deploying your code, the process is actually more important than the actual pressing of the button at the end of the day—or a thousand times a day for that matter. Before you can achieve this level of Continuous Delivery, you need to make sure that you aren't jeopardizing your system, users, and ultimately your business in the process.

Getting the code to production is where Continuous Delivery comes to complete the entire cycle from development to market.

This means that you need to know what makes the entire system work, where the potential problems are, and make sure that your deployment stack and new package take all this into consideration. To do so, you need to ensure that code is tested before deploying to production, and monitor code after deployment to confirm it's doing the right things. The "right things" mean that you are aware of the changes that are being made and how they affect the system as a whole.

Monitoring deployments is a whole manifesto unto itself, and it would be too lengthy to discuss this in detail. However, the focus on this is probably the most important factor, since the ability to deploy 1000 times a day is worthless if you aren't monitoring how these changes affect your system. Even though you're probably not playing to deploy that often, deploying a thousand times a day gives you the ability to fix things really fast and make small changes quickly.

That's why you need to have the entire process set up in such a way that enables you to quickly understand exactly what's going on in your system—and only then will you have the ability to deploy as many times as you want. This means monitoring the right places and

metrics; and it's just the tip of the iceberg. Everything has become exponentially more difficult these days with autoscaling capabilities, frequent changes to servers, and the locations of servers. Where once upon a time you had two servers and everything was simple, these days you have thousands of servers distributed around the world, and multiple processes to deploy all the time.

REACHING THE CONTINUOUS DELIVERY PROMISED LAND

When you're ready to deploy your code to production, you would need to write a process to ensure you take all aspects of the deployment into account; this would typically include:

- Choosing the right tool for the job—the tool isn't really the problem with deployment, the process just needs to be ironclad. That said, there are tools that are less deployment-oriented like common CM tools, where scripting tools may do a better job.
- Automating the process of pulling your server list—taking Amazon as an example, you can tag the different server types and then deploy to the right servers based on their type.
- Choosing the type of deployment process—there are a few common controlled deployment methods and there is much to be said about these.
- Monitoring the right things—so you can know when and what to rollback.

And while we can use these methods to deploy code, if we want to continuously deploy applications as a whole, including the infrastructure (infrastructure as code), then that's where orchestration would come in. We would still need to perform the same initial steps:

- Understanding the process and how it affects our system
- How to create the binaries/logical containers in a clean environment
- Testing and monitoring

However, on top of all of this, we could add code that creates infrastructure, including:

- Loading the resources
- Keeping the infrastructure in source control
- Deploying binaries with the infrastructure when the deployment is run

Whether you choose to do a canary or blue-green deployment, an orchestrator will come in handy in either scenario to manage business continuity and data integrity. With the complexity involved in pushing code to live systems in production, the orchestrator should be built in a manner that enables you to address the entire application lifecycle; a good example for this is TOSCA (Topology and Orchestration Specification for Cloud Applications). TOSCA is a YAML-based open cloud standard language from the OASIS organization (the creators of XML).

TOSCA has the combination of declarative descriptions of the application topology with all its components—including the load balancer, network, the compute resources, software, and everything else, along with an imperative set of workflows to describe the logic of any process you need to automate. What this means from a Continuous Delivery perspective is that each application component has lifecycle hooks that enable the adding of more hooks to cope with new processes, such as the invocation of A/B testing of deployments, testing, and monitoring.

Using containers is similar to VMs, but more lightweight... containers can be spawned pretty much instantly.

With these new CI/CD capabilities, a line of business that once had to request a feature from engineering based on business-level requirements, and then spend a year waiting for these to come to market, organizations can now expect a new feature to be shipped to production within just a few weeks. Needless to say, the business impact of such processes is driving an unprecedented evolution in IT, one that will only progress and gain momentum in the near future. However, as with all new technology, Continuous Delivery needs to be implemented with safe measures while taking all of the complexities into account, as the negative business impact of CD gone wrong can outweigh the positive aspects.

YARON PARASOL *is the VP Product Management at GigaSpaces, managing Cloudify. He has more than seven years of experience in enterprise software product management with special interest in IT management and monitoring, DevOps, cloud computing as well as in-memory and grid computing. You can follow Yaron on Twitter or Github – or find him in the park walking his beloved dog, Jeremy.*

"Ansible Tower has allowed us to provide better operations and security to our clients. It has also increased our efficiency as a team."

NASA uses Ansible Tower to centralize and control their Ansible automation initiative. With a real-time dashboard, role-based access control, credentials security, job scheduling, graphical inventory management and more, Ansible Tower is the best way to run Ansible in your organization, too.

Try Ansible Tower for free at ***ansible.com/tower***

ANSIBLE

Continuous Delivery Maturity Checklist

Check the boxes next to the practices you currently perform to see your maturity in each area of Continuous Delivery. Add up your score at the end based on the highest levels you checked.

Source Control

BASELINE
- [ ] Early branching
- [ ] Branches tend to remain apart

NOVICE
- [ ] Branches are used for isolating work
- [ ] Merges are common

INTERMEDIATE
- [ ] Pre-tested commits
- [ ] Integration branch is pristine

ADVANCED
- [ ] All commits are tied to tasks
- [ ] History used to rewrite features before pushing to central repository
- [ ] Version control DB schema changes

EXPERT
- [ ] Traceability analysis and release notes auto-generated
- [ ] Commits are clean enough for the master branch/trunk

Testing & QA

BASELINE
- [ ] Automatic unit testing with every build
- [ ] Code coverage is measured

NOVICE
- [ ] Peer-reviews
- [ ] Mockups & proxies used

INTERMEDIATE
- [ ] Periodic static code analysis
- [ ] Automated functional testing

ADVANCED
- [ ] Integrated management and maintenance of the test data
- [ ] Automated performance and security tests in target environments

EXPERT
- [ ] Automated acceptance testing

Build Process

BASELINE
- [ ] Official builds are not performed on developers' machines
- [ ] Self-service build or nightly build

NOVICE
- [ ] System polls source control and builds on commit
- [ ] Build artifacts are managed, some manual scripts still used

INTERMEDIATE
- [ ] Build artifacts are managed by purpose-built tools, no manual scripts
- [ ] Dependencies are managed in a repository

ADVANCED
- [ ] Distributed builds on build cluster, can be done in sequence
- [ ] Source control tells system when to build, no polling

EXPERT
- [ ] Build environments based on VMs
- [ ] Streams are never "broken"
- [ ] Gated commits

Visibility

BASELINE
- [ ] Build status notification is sent to committer

NOVICE
- [ ] Latest build status is available to all team members

INTERMEDIATE
- [ ] Trend reports are automatically generated from build server events
- [ ] People outside the team can subscribe to build statuses

ADVANCED
- [ ] Stakeholders have dashboards with real-time product and dependency stats

EXPERT
- [ ] Cross-team data mining and analysis

Deployment

BASELINE
- [ ] Fully scripted deployments

NOVICE
- [ ] Push-button deployments to test environments

INTERMEDIATE
- [ ] Auto deploy to first test environment
- [ ] Standard deployments across all environments
- [ ] Push-button deployments to production

ADVANCED
- [ ] Automated deployments after tests pass
- [ ] Database deployments
- [ ] Multi-tier deployments

EXPERT
- [ ] Ability to implement continuous deployment

Overall Maturity Scorecard

For each check mark add the assigned number of points and total them for each section:

POINT KEY:
BASELINE 0 PTS
NOVICE 1 PT
INTERMEDIATE 2 PTS
ADVANCED 3 PTS
EXPERT 4 PTS

TALLY YOUR SCORES:

Source Control	
Build Process	
Testing & QA	
Deployment	
Visibility	
TOTAL	

0-7 ADEQUATE **8-11** AVERAGE **12-14** SKILLED
15-17 ADEPT **18-20** MASTER

INSPIRED BY CHRIS SHAYAN AND ERIC MINICK

SOURCES: CHRISSHAYAN.ATLASSIAN.NET/WIKI/DISPLAY/MY/2013/07/23/CONTINUOUS+DELIVERY+MATRIX
IBMDW.NET/URBANCODE/DOCS/CONTINUOUS-DELIVERY-MATURITY-MODEL/

Is 2015 the year of continuous delivery for the database?

By Ben Rees

Have you seen the survey results in this year's Continuous Delivery Guide?

You probably weren't surprised to see uptake of continuous delivery for databases is half that for applications: 30% vs. 60%.

But these numbers hide an interesting nugget – last year, just 21% were doing it. That's 9% growth, in just one year, of continuous delivery for the database.

What's the cause? The barriers that prevented continuous delivery for the database are disappearing. Two, in particular:

1. **Team culture:** with the growth of DevOps, divisions are fading between development teams, tasked with delivering changes to customers quickly, and operations teams responsible for protecting business data at all costs.
2. **Tooling:** there's now tooling that makes databases first-class citizens in your continuous delivery process. It often integrates with existing CD infrastructure, such as build servers and version control systems, which makes adoption easier.

Of course, there are still challenges. If you've got a back-end database with business-critical data, it still needs to be preserved. This involves additional, database-specific considerations. Changing team culture isn't an overnight job, either.

"IT LITERALLY TAKES A FEW CLICKS TO DEPLOY TO AN ENVIRONMENT."

But we're seeing a movement emerge to meet these challenges: Database Lifecycle Management (DLM).

DLM encourages teams to use processes and technologies familiar from application CD to manage database changes. It's the same pathway, but with a few more stops for things like data migration, monitoring, and recovery.

Otherwise, the steps are the same. They need to be, because it's best practice to coordinate application and database development as part of a CD process wherever possible.

Version control

Version controlling database changes, for example, is a natural step for database development teams to communicate their changes. It promotes transparent development, provides a version to roll back to if required, and maintains a solid audit trail. Versioning database objects with application code also provides a single source of truth for development and deployment processes.

Continuous integration

Once database changes have been checked into version control, CI can be triggered. Teams can configure their build server to automatically pick up changes. Via a series of build steps, this will then run a series of tests which have been committed to version control.

Release management

Release management has been made simple for DBAs with a range of tools that deploy database changes alongside application changes. Some tools even allow DBAs to review changes and check that there's been no drift in production environments before they deploy.

Monitoring

After database changes have been deployed to production, it's important to continue monitoring for unexpected results. Once again, tools are out there that make monitoring so efficient that issues are highlighted before they cause a problem.

Recovery

The final step is to take a backup prior to deployment and to have a rollback and recovery strategy in place. That way, if there's a fault in the development process, your database can be restored to its original state.

Tooling is available to support these processes, including those made by Redgate. They integrate with software you probably already use for application CD, such as Git, Jenkins, and Bamboo.

Real users are feeling the benefit, in terms of time, effort, and money saved. Here's what one of ours, the IT Operations Manager at a property management firm had to say:

> "Database updates sometimes took a full working day to deploy. It cost us in excess of £400 [$600] to release a small update to our software.
>
> "Redgate tools enable us to reduce that time to around 15 minutes, with virtually no associated risk to live services and an extremely easy rollback process.
>
> "It literally takes a few clicks to deploy to an environment. We now have a process that costs tens of pounds, instead of hundreds."

Imagine deploying like that hundreds of times a year. How many hours and thousands of dollars would you save? Isn't that worth thinking about for 2015?

Ben Rees manages the database lifecycle management team at Redgate Software, using his experience in product and marketing management, plus five years as a developer in the trenches (in SQL Server, Oracle, and DB2), to make sure that in the growth of continuous delivery, the database isn't forgotten.

www.red-gate.com/delivery

redgate

ingeniously simple

Solutions Directory

Notice to readers: The data in this section is impartial and not influenced by any vendor.

This directory of configuration management, continuous integration, application release automation, and browser testing tools provides comprehensive, factual comparisons of data gathered from third-party sources and the tool creators' organizations. Solutions in the directory are selected based on several impartial criteria, including solution maturity, technical innovativeness, relevance, and data availability. The solution summaries underneath the product titles are based on the organization's opinion of its most distinguishing features. Understand that not having certain features is a positive thing in some cases. Fewer unnecessary features sometimes translates into more flexibility and better performance.

NOTE: The bulk of information gathered about these solutions is not present in these quarter-page profiles.

To view an extended profile of any product, simply go to: dzone.com/research/continuousdelivery

RELEASE AUTOMATION

Agile Software Factory by Grid Dynamics

Grid Dynamics' Agile Software Factory allows users to quickly build and release new software using iterative processes and prevent production defects.

MODELS
Agentless model

SUPPORTED PLATFORMS
- None

CI INTEGRATIONS
- Jenkins/Hudson

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
Does not support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

TWITTER @griddynamics

WEBSITE griddynamics.com/agile-software-factory

CONFIGURATION MANAGEMENT RESEARCH PARTNER

Ansible Tower by Ansible

Ansible Tower is a browser-based UI with a fully discoverable API that features role-based access control, job scheduling, auditing and out-of-the-box integrations with AWS, GCE, VMware, Azure and Rackspace.

MODELS
Agentless model

HOSTING
On-premise

COMMUNICATION PROTOCOL
- SSH

DATA STORES
- PostgreSQL

FEATURES
- Has role-based access control
- Configurations stored in source control
- Integrates with LDAP for identity

WEB UI FEATURES
- Ability to browse a repository of available configurations
- Ability to view the current configuration version applied to machines
- Ability to trigger configuration of one or more machines
- Ability to show any difference between the actual and assumed configuration setting of a machine

CONFIG. DESCRIPTION
- Text-style (YAML, JSON)

TWITTER @ansible

WEBSITE ansible.com

CONFIGURATION MANAGEMENT | RESEARCH PARTNER

Ansible Open Source by Ansible

Ansible is an open source IT automation and orchestration engine. It can handle configuration management, application deployment and orchestration of complex IT workflows.

MODELS
Agentless model

HOSTING
On-premise

COMMUNICATION PROTOCOLS
- SSH
- Powershell

DATA STORES
- Decentralized

FEATURES
- Free and open source
- Agentless model reduces complexity of managing systems
- Simple, self-documenting YAML-based automation language
- One-tool achieves configuration management & orchestration

CONFIG. DESCRIPTION
- Text-style (YAML, JSON)

TWITTER @ansible
WEBSITE ansible.com

CONTINUOUS INTEGRATION

Bamboo by Atlassian

Atlassian Bamboo can automatically generate new build plans for new branches with main line's CI scheme, and can trigger deployments when a test run completes, on a schedule, or ad-hoc.

MODELS
Push or pull agent or agentless model

SUPPORTED PLATFORMS
- Java Runtime Environment

CM INTEGRATIONS
- Puppet

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
- Does support 3rd-party plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments
- Automatic validation builds for code changes to the master branch

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites.

TWITTER @atldevtools
WEBSITE atlassian.com/software/bamboo

RELEASE AUTOMATION

BMC Release Lifecycle Management by BMC Software

BMC Release Lifecycle Management approaches application deployment from an application perspective, enabling teams to automate and coordinate processes across the application release cycle.

MODELS
Push agent model

SUPPORTED PLATFORMS
- Java Runtime Environment
- .NET Environment
- Ruby Environment
- PHP Environment

CI INTEGRATIONS
- Jenkins/Hudson

HOSTING
On-premise

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

WEBSITE bmc.com/it-solutions/release-process-management.html

BROWSER TESTING

BrowserStack Automate by BrowserStack

BrowserStack Automate allows users to automatically test several platforms and browsers, with unlimited testing time and live debugging capabilities.

OPEN SOURCE
No

CI TOOL SUPPORT
- Jenkins
- Travis CI
- CircleCI

TEST SUPPORT
- Cross-browser testing
- Mobile testing
- Selenium
- JavaScript testing
- Manual testing

FEATURES
- Detailed session history and logs
- Test local and internal servers
- Live automated test video
- Test screenshot functionality
- Secure cloud testing environments
- Test native and hybrid mobile apps
- Automated and real-time testing

PRICING
Subscription-based

TWITTER @browserstack
WEBSITE browserstack.com

CONTINUOUS INTEGRATION

Buildkite by Buildkite

Buildkite is a simple, open source agent that provides self-hosted continuous integration and delivery, and can run code from SCMs like GitHub.

MODELS
Push agents

SUPPORTED PLATFORMS
- Java Runtime Environment
- .NET Environment
- Ruby Environment
- PHP Environment

CM INTEGRATIONS
- None

HOSTING
On-premise

THIRD PARTY PLUGINS
- Does not support 3rd-party plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments
- Automatic validation builds for code changes to the master branch

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites.

TWITTER @buildkite
WEBSITE buildkite.com

RELEASE AUTOMATION

Buildmaster by Inedo

Inedo Buildmaster is designed for use by entire team to manage continous delivery, and supports complex scenarios from parallel deployments to server pools.

MODELS
Push agent or agentless model

SUPPORTED PLATFORMS
- .NET Environment

CI INTEGRATIONS
- Jenkins/Hudson
- Atlassian Bamboo
- JetBrains TeamCity
- Microsoft Team Foundation Server

HOSTING
On-premise

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email
- RSS
- Chat (XMPP, MSN, IRC, GTalk)
- SMS/Text Message
- HipChat
- Webhooks
- Remote API
- SOAP
- Twitter

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

TWITTER @inedo
WEBSITE inedo.com/buildmaster

RELEASE AUTOMATION RESEARCH PARTNER

CA Release Automation by CA Technologies

CA Release Automation provides sharable components that enable the reuse of workflows across applications, and a graphical design of processes without scripting.

MODELS
Push or pull agent model

SUPPORTED PLATFORMS
- Java Runtime Environment

CI INTEGRATIONS
- Jenkins/Hudson
- JetBrains TeamCity
- Microsoft Team Foundation Server

HOSTING
On-premise

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites.

TWITTER @cainc
WEBSITE ca.com/release-automation

CONFIGURATION MANAGEMENT

CFEngine Community by CFEngine

CFEngine is an IT automation service that provides visibility and management through self-service, alerts, and event logs, along with reporting functionality.

MODELS
Pull agent model

HOSTING
Cloud-hosted or on-premise

COMMUNICATION PROTOCOL
- CFEngine proprietary protocol on port TCP/5308

DATA STORES
- PostgreSQL

FEATURES
- Has role-based access control
- Configurations stored in source control
- Integrates with Active Directory and LDAP for identity

WEB UI FEATURES
- Ability to browse a repository of available configurations
- Ability to view the current configuration version applied to machines
- Ability to trigger configuration of one or more machines
- Ability to show the actual configuration settings of a current machine (i.e. whatÕs really configured on the machine right now)
- Ability to show any difference between the actual and assumed configuration setting of a machine
- Ability to show machine data (OS, Memory, CPU, etc.)

CONFIG. DESCRIPTION
- Text-style (YAML, JSON)
- DSL-style

TWITTER @cfengine
WEBSITE cfengine.com

CONFIGURATION MANAGEMENT

Chef by Chef

Chef is an IT automation service that relies on reusable definitions known as recipes to automate infrastructure tasks.

MODELS
Push or pull agent model

HOSTING
On-premise

COMMUNICATION PROTOCOL
- HTTPS

DATA STORES
- PostgreSQL
- CouchDB
- Solr

FEATURES
- Has role-based access control

WEB UI FEATURES
- Ability to browse a repository of available configurations
- Ability to view the current configuration version applied to machines
- Ability to trigger configuration of one or more machines
- Ability to show machine data

CONFIG. DESCRIPTION
- Text-style (YAML, JSON)

TWITTER @chef
WEBSITE chef.io

CONTINUOUS INTEGRATION

CircleCI by CircleCI

CircleCI is a cloud-hosted service with parallelism across machines with intelligent test splitting, and offers tight integration with GitHub, including the PR status API.

MODELS
Agentless

SUPPORTED PLATFORMS
- Java Runtime Environment
- Ruby Environment
- PHP Environment
- Python Environment

CM INTEGRATIONS
- Doable with simple script

HOSTING
Cloud-hosted

THIRD PARTY PLUGINS
- Does not support 3rd-party plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments
- Automatic validation builds for code changes to the master branch
- Ability to change the default configuration/template for the standard build machines

REQUIRED SOURCE CONTROL WEBSITES
Requires source control through GitHub and BitBucket

TWITTER @circleci
WEBSITE circleci.com

CONTINUOUS INTEGRATION

Codeship by Codeship

Codeship offers strong integration with many deployment providers and has fully web-based configuration of all build steps.

MODELS
Agentless

SUPPORTED PLATFORMS
- Java Runtime Environment
- Ruby Environment
- PHP Environment
- Python Environment

CM INTEGRATIONS
- None

HOSTING
Cloud-hosted

THIRD PARTY PLUGINS
- Does not support 3rd-party plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments
- Automatic validation builds for code changes to the master branch

REQUIRED SOURCE CONTROL WEBSITES
Requires source control through GitHub and BitBucket

TWITTER @codeship
WEBSITE codeship.com

CONTINUOUS INTEGRATION

drone.io by drone.io

drone.io is a continuous integration solution that virtualizes every build in a Docker container, with several resources to help developers use Drone.io with most languages.

MODELS
Agentless

SUPPORTED PLATFORMS
- Java Runtime Environment
- Ruby Environment
- PHP Environment
- Python Environment

CM INTEGRATIONS
- None

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
- Does not support 3rd-party plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments
- Automatic validation builds for code changes to the master branch

REQUIRED SOURCE CONTROL WEBSITES
Requires source control through GitHub and BitBucket

TWITTER @droneio
WEBSITE drone.io

RELEASE AUTOMATION

ElectricFlow Deploy by Electric Cloud

ElectricFlow Deploy features flexible model-driven environment provisioning and application deployment, and provides shared control and visibility into several toolchains.

MODELS
Push or pull agent model

SUPPORTED PLATFORMS
- Java Runtime Environment
- .NET Environment
- Ruby Environment
- PHP Environment

CI INTEGRATIONS
- Jenkins/Hudson

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email
- RSS
- Chat
- SMS/Text Message
- HipChat
- Campfire
- Webhooks
- Remote API
- SOAP
- Twitter

REQUIRED SOURCE CONTROL WEBSITES
Does require source control with Github or SourceForge

TWITTER @electriccloud

WEBSITE electric-cloud.com/deploy

RELEASE AUTOMATION

Gradle by Gradleware

Gradle is an open source release automation project that allows users to establish any enterprise policy across multi-project, multi-platform builds.

MODELS
Push or Pull agent or agentless model

SUPPORTED PLATFORMS
- Java Runtime Environment

CI INTEGRATIONS
- Jenkins/Hudson
- Atlassian Bamboo
- TravisCI
- JetBrains TeamCity
- Electric Cloud

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email
- RSS
- Chat (XMPP, MSN, IRC, GTalk)
- SMS/Text Message
- HipChat
- Campfire
- Webhooks
- Remote API
- SOAP
- Twitter

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

TWITTER @gradleware

WEBSITE gradle.org

CONTINUOUS INTEGRATION

Hudson Continuous Integration Server by Eclipse Foundation

Eclipse's Hudson Continuous Integration Server is a self-contained, Java-based solution that runs on a wide range of standard environments.

MODELS
Push agent or agentless model

SUPPORTED PLATFORMS
- Java Runtime Environment

CM INTEGRATIONS
- None

HOSTING
On-premise

THIRD PARTY PLUGINS
- Does support 3rd-party plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites.

TWITTER @hudsonci

WEBSITE eclipse.org/hudson

CONTINUOUS INTEGRATION RESEARCH PARTNER

Jenkins Enterprise by CloudBees

CloudBees Jenkins Enterprise gives users Continuous Delivery with Jenkins at the Enterprise scale and offers mission critical support for Jenkins and all plugins.

MODELS
Push or pull agents

SUPPORTED PLATFORMS
- Java Runtime Environment

CM INTEGRATIONS
- Chef
- Puppet
- SaltStack
- Ansible
- cfengine

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
- Does support 3rd-party plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments
- Automatic validation builds for code changes to the master branch

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites.

TWITTER @cloudbees

WEBSITE cloudbees.com

RELEASE AUTOMATION

Leroy by Epic Force

Epic Force Leroy allows for native, cross-platform support and offers a free-to-use option for legal use.

MODELS
Pull agent model

SUPPORTED PLATFORMS
- Native

CI INTEGRATIONS
- Jenkins/Hudson

HOSTING
On-premise

THIRD PARTY PLUGINS
Does not support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

TWITTER @leroydeploy
WEBSITE leroydeploy.com

RELEASE AUTOMATION

ONE Automation by Automic

Automic ONE Automation provides automatic rollback, deploys through managed file transfer, and uses the same workflow in every environment.

MODELS
Push or pull agent or agentless model

SUPPORTED PLATFORMS
- Java Runtime Environment
- .NET Environment
- Ruby Environment
- PHP Environment

CI INTEGRATIONS
- Jenkins/Hudson
- Atlassian Bamboo
- JetBrains TeamCity
- Microsoft Team Foundation Server

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email
- RSS
- Chat
- SMS/Text Message
- Webhooks
- Remote API
- SOAP
- Twitter

REQUIRED SOURCE CONTROL WEBSITES
Does require source control with Github or SourceForge

TWITTER @automic
WEBSITE automic.com

RELEASE AUTOMATION

Plutora by Plutora

Plutora's collaborative editor enables multiple participants to construct runsheets interactively and allocate deployment tasks to internal delivery teams and external suppliers.

MODELS
Agentless model

SUPPORTED PLATFORMS
- None

CI INTEGRATIONS
- Jenkins/Hudson
- Atlassian Bamboo
- Microsoft Team Foundation Server

HOSTING
Cloud-hosted

THIRD PARTY PLUGINS
Only vendor plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email
- RSS
- Chat (XMPP, MSN, IRC, GTalk)
- SMS/Text Message
- HipChat
- Twitter

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

TWITTER @plutora
WEBSITE plutora.com

CONFIGURATION MANAGEMENT

Puppet Enterprise by Puppet Labs

Puppet Enterprise automates tasks at many stages of the IT infrastructure lifecycle, including: discovery, provisioning, configuration management, orchestration, and reporting.

MODELS
Push or pull agent or agentless model

HOSTING
Cloud-hosted or On-premise

COMMUNICATION PROTOCOL
- SSH
- HTTP

DATA STORES
- PostgreSQL
- PuppetDB
- Hiera

FEATURES
- Has role-based access control
- Configurations stored in source control
- Integrates with Active Directory and LDAP for identity

WEB UI FEATURES
- Ability to view the current configuration version applied to machines x Ability to trigger configuration of one or more machines
- Ability to show the actual configuration settings of a current machine (i.e. what's really configured on the machine right now)
- Ability to show any difference between the actual and assumed configuration setting of a machine
- Ability to show machine data (OS, Memory, CPU, etc.)

CONFIG. DESCRIPTION
- DSL-style
- Programming language-style

TWITTER @puppetlabs
WEBSITE puppetlabs.com

RELEASE AUTOMATION

RapidDeploy by MidVision

MidVision RapidDeploy is an open and flexible platform for orchestrating development and operations, with connection to any target, toolchain, cloud, or transport.

MODELS
Push agent model

SUPPORTED PLATFORMS
- Java Runtime Environment

CI INTEGRATIONS
- Jenkins/Hudson
- Atlassian Bamboo
- Microsoft Team Foundation Server

HOSTING
On-premise

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
On success and on failure

NOTIFICATIONS SUPPORTED
- Email
- RSS
- SMS/Text Message
- Webhooks
- Remote API

REQUIRED SOURCE CONTROL WEBSITES
Does require source control with Github or BitBucket

TWITTER @midvision

WEBSITE midvision.com

RELEASE AUTOMATION

RepliWeb by Attunity

Attunity RepliWeb is a content and code deployment platform for organizations that need to deploy applications and large-file content throughout distributed datacenters and cloud infrastructure.

MODELS
Push or pull agent model

SUPPORTED PLATFORMS
- .NET Environment

CI INTEGRATIONS
- Microsoft Team Foundation Server

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
Does not support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

TWITTER @attunity

WEBSITE attunity.com

RELEASE AUTOMATION

Rocket ALM Hub by Rocket Software

Rocket ALM Hub provides multiplatform application lifecycle management automation from the time a request for change arises until the solution is delivered.

MODELS
Push or pull agent model

SUPPORTED PLATFORMS
- Java Runtime Environment
- .NET Environment
- PHP Environment
- Python Environmnent
- RPG environment

CI INTEGRATIONS
- Jenkins/Hudson

HOSTING
On-premise

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

TWITTER @Rocket

WEBSITE rocketsoftware.com

CONFIGURATION MANAGEMENT

SaltStack Enterprise by SaltStack

SaltStack provides fast, scalable and flexible systems, and configuration management software for CloudOps, ITOps and DevOps.

MODELS
Push or pull agent or agentless model

HOSTING
Cloud-hosted

COMMUNICATION PROTOCOL
- SSH
- HTTP
- ZeroMQ
- UDP

DATA STORES
- MySQL
- PostgreSQL
- MongoDB
- CouchDB
- More

FEATURES
- Has role-based access control
- Configurations stored in source control
- Integrates with Active Directory and LDAP for identity

WEB UI FEATURES
- Ability to view the current configuration version applied to machines
- Ability to trigger configuration of one or more machines
- Ability to show the actual configuration settings of a current machine)
- Ability to show any difference between the actual and assumed configuration setting of a machine
- Ability to show machine data

CONFIG. DESCRIPTION
- Text-style (YAML, JSON)
- DSL-style
- Programming language-style (Ruby)

TWITTER @saltstackinc

WEBSITE saltstack.com

FUNCTIONAL AND UNIT TESTING | RESEARCH PARTNER

Sauce Labs Platform by Sauce Labs

Sauce Labs provides a complete testing platform for native, hybrid, and web apps, allowing users to run Selenium, Appium, JS unit, and manual tests in any language on 450+ browser and platform configurations.

OPEN SOURCE
Yes

CI TOOL SUPPORT
- Jenkins
- TravisCI
- CircleCI
- Bamboo
- TeamCity

TEST SUPPORT
- Cross-browser testing
- Mobile testing
- Selenium
- Appium
- JavaScript testing
- Manual testing

FEATURES
- Detailed session history and logs
- Parallel test execution
- Test local and internal servers
- Live automated test video
- Secure local tunneling
- Test screenshot functionality

PRICING
By number of VMs and test run minutes

TWITTER @saucelabs
WEBSITE saucelabs.com

CONTINUOUS INTEGRATION

Semaphore by Rendered Text

Semaphore provides automatic project configuration without any change in source code, and VM-level build parallelization in continuous deployment pipelines.

MODELS
Agentless

SUPPORTED PLATFORMS
- Java Runtime Environment
- Ruby Environment
- PHP Environment
- Python Environment

CM INTEGRATIONS
- Chef

HOSTING
Cloud-hosted

THIRD PARTY PLUGINS
- Only vendor plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments
- Automatic validation builds for code changes to the master branch
- Ability to change the default configuration/template for the standard build machines

REQUIRED SOURCE CONTROL WEBSITES
Requires source control through GitHub and BitBucket

TWITTER @semaphoreapp
WEBSITE semaphoreapp.com

RELEASE AUTOMATION

Serena Deployment Automation by Serena Software

Serena Deployment Automation offers a built in pipeline with graphical modelling of optional or mandatory pipeline views, and supports mobile deployments and monitoring.

MODELS
Push or pull agent model

SUPPORTED PLATFORMS
- Java Runtime Environment
- .NET Environment
- Ruby Environment
- PHP Environment

CI INTEGRATIONS
- Jenkins/Hudson
- Atlassian Bamboo
- JetBrains TeamCity
- Microsoft Team Foundation Server

HOSTING
On-premise

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email
- RSS
- SOAP
- ALF

REQUIRED SOURCE CONTROL WEBSITES
Does require source control with Github

TWITTER @serenasoftware
WEBSITE serena.com/sda

CONTINUOUS INTEGRATION

Shippable by Shippable

Users can use Shippable to trigger multiple builds to test language runtimes and environments and migrate their legacy CI system.

MODELS
Pull agent or agentless models

SUPPORTED PLATFORMS
- Java Runtime Environment
- Ruby Environment
- PHP Environment
- Python Environment

CM INTEGRATIONS
- SaltStack
- Ansible

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
- Does support 3rd-party plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments
- Version tracking for application configuration and environment properties
- Automatic validation builds for code changes to the master branch

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites.

TWITTER @beshippable
WEBSITE shippable.com

CONTINUOUS INTEGRATION | RESEARCH PARTNER

Snap CI by Thoughtworks

Snap CI is a continuous deployment and integration tool with automatic test parallelization and branch tracking builds and pull-request support. It also features first class support for deployment pipelines and simple in-browser build debugging.

MODELS
Agentless

SUPPORTED PLATFORMS
- None

CM INTEGRATIONS
- None

HOSTING
Cloud-hosted

THIRD PARTY PLUGINS
- Not supported

FEATURES
- Sets new builds to be the trigger for new deployments
- Automatic validation builds for code changes to the master branch
- Ability to change the default configuration/template for the standard build machines

REQUIRED SOURCE CONTROL WEBSITES
Requires source control through GitHub

TWITTER @snap_ci
WEBSITE snap-ci.com

CONTINUOUS INTEGRATION

Solano CI by Solano Labs

Solano CI provides fully-isolated, reproducible builds for more effective debug and regression handling while being able to dynamically scale to several cores on demand.

MODELS
Push or pull agent or agentless model

SUPPORTED PLATFORMS
- Java Runtime Environment
- Ruby Environment
- PHP Environment
- Python Environment

CM INTEGRATIONS
- None

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
- Does not support 3rd-party plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments
- Automatic validation builds for code changes to the master branch
- Ability to change the default configuration/template for the standard build machines

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites.

TWITTER @solanolabs
WEBSITE solanolabs.com

CONFIGURATION MANAGEMENT | RESEARCH PARTNER

Stackato by ActiveState

Stackato is a commercially supported PaaS that leverages open source components such as Cloud Foundry and Docker, and integrates with Jenkins, Bamboo and other CI systems. It runs on top of any cloud as a self-service platform for deploying, updating and managing applications.

MODELS
Push or pull agent or agentless model

HOSTING
Cloud-hosted or on-premise

COMMUNICATION PROTOCOL
- SSH
- HTTP

DATA STORES
- MySQL
- PostgreSQL
- MongoDB
- PuppetDB
- Redis

FEATURES
- Automatically provisions databases and services
- Can deploy apps directly from CI system, IDE or CLI client
- Easy management of app instances and environments

WEB UI FEATURES
- Ability to view the current configuration version applied to machines
- Ability to trigger configuration of one or more machines
- Ability to show the actual configuration settings of a current machine (i.e. what's really configured on the machine right now)
- Ability to show any difference between the actual and assumed configuration setting of a machine
- Ability to show machine data (OS, Memory, CPU, etc.)

CONFIG. DESCRIPTION
- Text-style (YAML, JSON)
- Programming language-style (Ruby)

TWITTER @activestate
WEBSITE activestate.com/stackato

CONTINUOUS INTEGRATION

TeamCity by JetBrains

JetBrains TeamCity features reusable setting templates, heirarchical projects, parameter references, and comes bundled with other JetBrains tools like IntelliJ IDEA.

MODELS
Push or pull agents

SUPPORTED PLATFORMS
- Java Runtime Environment
- NET Environment
- Ruby Environment

CM INTEGRATIONS
- None

HOSTING
On-premise

THIRD PARTY PLUGINS
- Does support 3rd-party plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments
- Automatic validation builds for code changes to the master branch
- Version tracking for application configuration and environment properties

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites.

TWITTER @teamcity
WEBSITE jetbrains.com/teamcity

CONTINUOUS INTEGRATION

Travis CI by Travis CI

Travis CI is an open-source solution with deep integration and collaboration with GitHub and an expansive community.

MODELS
Push agents

SUPPORTED PLATFORMS
- Java Runtime Environment
- Ruby Environment

CM INTEGRATIONS
- None

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
- Does not support 3rd-party plugins

FEATURES
- Designates source control check-in as the mechanism to trigger a deployment
- Sets new builds to be the trigger for new deployments
- Version tracking for application configuration and einvironment properties
- Automatic validation builds for code changes to the master branch
- Ability to change the default configuration/template for the standard build machines

REQUIRED SOURCE CONTROL WEBSITES
Requires source control through Github

TWITTER @travisci
WEBSITE travis-ci.com

RELEASE AUTOMATION

UrbanCode Deploy with Patterns by IBM

IBM UrbanCode features an open integration plugin framework to minimizes scripting, and combines service versions, database changes, and configuration changes into a single scalable application.

MODELS
Push agent model

SUPPORTED PLATFORMS
- Java Runtime Environment

CI INTEGRATIONS
- Jenkins/Hudson
- JetBrains TeamCity
- Microsoft Team Foundation Server

HOSTING
On-premise

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
On success and on failure

NOTIFICATIONS SUPPORTED
- Email
- Remote API

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

TWITTER @urbancode
WEBSITE developer.ibm.com/urbancode/products/urbancode-deploy

RELEASE AUTOMATION

XL Deploy by Xebia Labs

XL Deploy features a fully automated deployment plan generation tool based on configurable rules, without scripting.

MODELS
Agentless model

SUPPORTED PLATFORMS
- Java Runtime Environment

CI INTEGRATIONS
- Jenkins/Hudson
- Atlassian Bamboo
- Microsoft Team Foundation Server

HOSTING
On-premise

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

TWITTER @xebialabs
WEBSITE xebialabs.com

RELEASE AUTOMATION

Zend Server by Zend

Zend Server is focused on bringing continuous deployment to PHP applications, and includes monitoring and configuration management capabilities.

MODELS
Agentless

SUPPORTED PLATFORMS
- PHP Environment

CI INTEGRATIONS
- Jenkins/Hudson
- Atlassian Bamboo

HOSTING
On-premise or cloud-hosted

THIRD PARTY PLUGINS
Does support 3rd-party plugins

NOTIFICATION DELIVERY
Any custom step in the process

NOTIFICATIONS SUPPORTED
- Email
- Remote API

REQUIRED SOURCE CONTROL WEBSITES
Does not require source control through websites

TWITTER @zend
WEBSITE zend.com

glossary

a

Agent
A program installed on specific physical servers in order to handle the execution of various processes on that server.

Agile Software Development
An iterative and incremental software development workflow methodology with several sub-methodologies.

Application Release Automation (ARA)
A type of continuous software release solution that handles packaging and deploying applications into a variety of environments and ultimately into production.

b

Behavior-Driven Development (BDD)
A development methodology that asserts software should be specified in terms of the desired behavior of the application, and with syntax that is readable for business managers.

Build Agent
A type of agent used in continuous integration that can be installed locally or remotely in relation to the continuous integration server. It sends and receives messages about handling software builds.

Build Artifact Repository
A tool used to organize artifacts with metadata constructs and to allow automated publication and consumption of those artifacts.

c

Capacity Test
A test that is used to determine the maximum number of users a computer, server, or application can support just before failing.

Configuration Drift
A term for the general tendency of software and hardware configurations to drift, or become inconsistent, with the template version of the system due to manual ad hoc changes (like hotfixes) that are not introduced back into the template.

Configuration Management
A term for establishing and maintaining consistent settings and functional attributes for a system. It includes tools for system administration tasks such as IT infrastructure automation (e.g. Chef, Puppet, etc.).

Containerization
A server virtualization method where the kernel of an operating system allows for multiple isolated user space instances, instead of just one.

Continuous Delivery
A software production process where the software can be released to production at any time with as much automation as possible for each step.

Continuous Deployment
A software production process where changes are automatically deployed to production without any manual intervention.

Continuous Integration (CI)
A software development process where a continuous integration server rebuilds a branch of source code every time code is committed to the source control system. The process is often extended to include deployment, installation, and testing of applications in production environments.

d

Deployment
A term that refers to the grouping of every activity that makes a program available for use and moving it to the target environment.

Deployment Pipeline
An automated process for getting software from version control to the production environment by moving those builds through multiple stages of testing and deployment.

DevOps
An IT organizational methodology where all teams in the organization, especially development teams and operations teams, collaborate and implement technology to increase software production agility and achieve business goals.

e

Exploratory Testing
A manual testing strategy where human testers have the freedom to test areas where they suspect issues could arise that automated testing won't catch.

i

Infrastructure-as-a-Service (IaaS)
A self-service computing, networking, and storage utility on-demand over a network.

Integration Testing
Testing that occurs after unit testing, but before validation testing, where individual software components are combined and tested as a single group.

Issue Tracking
Also known as bug tracking or defect tracking, this is a process that allows programmers and quality assurance personnel to track the flow of defects and new features from identification to resolution.

m

Mean-Time-Between-Failures (MTBF)
An average time between failures, used to measure reliability of a system or component.

Mean-Time-to-Recovery (MTTR)
The average time to recovery a system or component from a failure and return it to production status.

p

Platform-as-a-Service (PaaS)
An application-server-as-a-service (aPaaS) provided over a network along with additional components that may include middleware, development tools, and application controllers.

Production
The final stage in a deployment pipeline where the software will be used by the intended audience.

r

Rollback
An automatic or manual operation that restores a database or program to a previous defined state in order to recover from an error.

s

Self-Service Deployment
A feature where deployment processes are automated enough for developers to allow project managers or even clients to directly control push-button deployments.

Source Control
A system for storing, tracking, and managing changes to software. This is commonly done through a process of creating branches (copies for safely creating new features) off of the stable master version of the software and then merging stable feature branches back into the master version. This is also known as version control or revision control.

u

Unit Testing
A testing strategy in which the smallest unit of testable code is isolated from the rest of the software and tested to determine if it functions properly.

User Acceptance Test
The final phase of software testing where clients and end-users determine whether the program will work for the end-user in real world scenarios. This stage is also known as beta testing.

v

Virtual Machine (VM)
A software emulation of a physical computing resource that can be modified independent of the hardware attributes.