Orchestrating the DevOps Tool Chain

Continuous delivery for the Enterprise

April 2015

Executive Briefing Guide

In association with

Foreword

What's the single most important change your business executives would highlight if you asked them how IT could improve?

Not so long ago, the most likely answer might have been "Spend less money", or "Deliver a more reliable service", but priorities are changing. Effective cost and service level management remain important, but against the backdrop of a fast moving digital marketplace, a new imperative has taken precedent. Today, executives are more likely to express wishes such as "Respond more quickly and flexibly" or "Be more proactive about helping us to create competitive advantage".

If you are involved in software delivery, the chances are that you have been impacted by the shifts these comments reflect, and are being asked to play your part in getting new capability into the hands of users more rapidly than ever.

In this paper, which was commissioned by CA Technologies, we explore how to create a robust, sustainable and inclusive software delivery environment built on 'DevOps' principles in order to better meet emerging challenges and expectations.

If you are familiar with DevOps already, our aim is to outline how typical first steps can be pulled together through higher-level orchestration of the delivery process to enable a more strategic enterprise-wide approach. If that last sentence didn't mean that much to you, don't worry because we'll review the key concepts as we go.

Inputs into the discussion

This paper is based on a combination of research, knowledge and insights from Freeform Dynamics, together with input from adopters of CA Technologies' Release Automation software gathered at a recent user group event.

Please note that while we make reference to CA solutions during our discussion, this is not to endorse them, but simply to illustrate what can be achieved through the use of modern technology and techniques in this space.

Contents

Introduction

Traditional priorities such as cost control and quality of service are still important, but attention is shifting to the imperatives of what some are calling the 'app economy'.

A great deal has been written about how trends in digital communication and engagement are changing the business landscape, and you probably don't need reminding of what's going on in this space. Suffice it to say that if your organisation hasn't been impacted by the way in which the web, mobile technology and social media are now embedded in society and business, then it's only a matter of time.

These digital market mechanics change what's expected of IT. Traditional priorities such as cost control and quality of service are still important, but attention is shifting to the imperatives of what some are calling the 'app economy'. These include a constant need to innovate and deliver fresh user experiences in order to keep customers interested and engaged, and, of course, to keep the competition at bay. Gone are the days when you could get away with 'good enough' capability updated once or twice a year.

Gone are the days when you could get away with 'good enough' capability updated once or twice a year.

Our discussion in this paper will examine some of the implications from a software development and deployment perspective. We'll specifically look at the need for next-generation tooling to enable rapid iteration within the software delivery cycle, along with higher level automation, visibility and control across the whole delivery process. The aim is to work around well-known limitations of the way most enterprises have designed, built and deployed software over the past three decades or more.

From waterfalls and silos to DevOps and continuous delivery

Traditional approaches to software delivery are not adequate to cope with today's digital business needs. The rigidity of old waterfall methods and the delays arising from disconnects between development and operational silos make it very hard to keep up. An alternative approach known as 'DevOps' is therefore gaining momentum.

The rigidity of old waterfall methods and the delays arising from disconnects between development and operational silos make it very hard to keep up.

Pulling together advances in areas such as agile development, software-defined datacentres and cloud computing, DevOps is about streamlining software delivery from requirements through to operations. The traditional disconnects between developers and operational staff are addressed along the way, and it's this that gives rise to the name. DevOps also embraces the principle of 'continuous delivery' aimed at creating deployable business value on a frequent and ongoing basis. Built-in feedback loops, including from the live environment right back to developers, are then in place to enable continuous tuning and optimisation.

DevOps is about streamlining software delivery from requirements to operations.

DevOps is a big topic and it's beyond the scope of this paper to explore it fully. The aspect on which we will focus from here on in is how you enable it from a tooling perspective. Within this we will touch on solutions that:

a) Help automate much of the activity within development and operations teams.

b) Bridge the gap between development and operations to allow rapid creation and refresh of development, testing, production and other environments.

c) Enable new and updated software to be promoted efficiently, reliably and consistently through the delivery pipeline, e.g. from the development environment, through test and staging, and ultimately to the live landscape.

The way you implement such capability will determine your level of success. Let's therefore walk through the considerations in a structured manner.

DevOps origins and evolution

The original thinking behind DevOps emerged from within the IT practitioner community, largely in direct response to the frustrations with traditional IT.

The original thinking behind DevOps emerged from within the IT practitioner community, largely in direct response to the frustrations with traditional IT delivery methods previously discussed. The ideas quickly gained traction among so-called 'born on the web' businesses, and since then DevOps has become instrumental in enabling many cloud service providers and mobile developers to roll out functionality on a continuous basis while maintaining a high degree of quality and operational integrity.

An unfortunate clash of cultures

DevOps adoption has until recently, however, been comparatively slow in mainstream large enterprise IT departments, despite the obvious need arising from digital business dynamics. Apart from inertia associated with the use of traditional methods, the way in which DevOps has grown up has often jarred with the historical enterprise emphasis on control and stability. The DevOps philosophy of cross functional integration has then frequently conflicted with deeply-ingrained lines of demarcation between teams.

It's easy to get the impression that to 'do DevOps properly' you need to shun all commercial software in favour of open source, and forget everything you thought you knew about IT delivery.

These cultural impediments have arguably been aggravated by perceptions of the 'DevOps movement'. When enthusiasts get together at conferences, for example, their passion and evangelism can sometimes be misinterpreted as elitism and purism. It's easy to get the impression that to 'do DevOps properly' you need to shun all commercial software in favour of open source, and forget everything you thought you knew about IT delivery because it has all been superseded. Those working in a complex legacy-dependent environment can find this off-putting.

Making DevOps work in the enterprise

The reality is that DevOps is completely relevant to and compatible with the requirements of enterprise IT.

The reality is that DevOps is completely relevant to and compatible with the requirements of enterprise IT, and indeed many blue-chip corporates, ranging from large retailers like Tesco, to major telcos such as Swisscom, have adopted it at a strategic level as we shall see later. In order to succeed, however, the enterprise must move some way towards the more revolutionary ideas and principles that underpin DevOps. Equally, though, some of the practical aspects of the DevOps approach need to be adjusted, extended or strengthened to deal with the needs of a highly complex, large-scale environment, respecting the fact that applications vary considerably in their risk profile and their 'need for speed' from a deployment perspective (Figure 1).

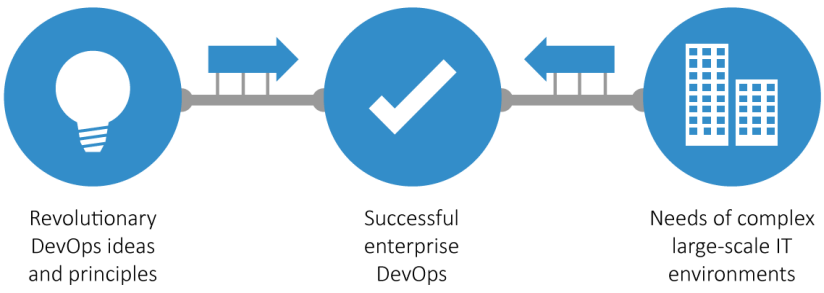

Figure 1

DevOps ideals meet enterprise IT reality

DevOps is about people, processes and technology.

When you drill into what this means in practice, just like any other major transformational activity, DevOps is about people, processes and technology. Our treatment in this paper is mostly concerned with the tools that enable the creation of a robust enterprise class DevOps environment (essentially the technology dimension), but we will cover relevant process and cultural considerations in context as we go.

Foundational automation

The natural evolution of activity within development and operations lays some good foundations for DevOps.

The good news is that the natural evolution of activity within development and operations lays some good foundations for DevOps. Understanding this is a good place to start when looking to take things forward.

Agile development and continuous integration

The chances are that at least one of your teams has adopted or is exploring agile development.

In response to the need for more speed and flexibility, the chances are that at least one of your teams has adopted or is exploring agile development techniques. The aim is to deliver new capability and value to the business in smaller chunks, but much more frequently. A number of well-defined methodologies exist that deal with the process and organisational dimensions of agile development, e.g. DSDM, SCRUM, Adaptive Software Development, and Extreme Programming.

At the heart of agile is a concept known as 'continuous integration' (CI). The idea is that developers, often from multiple teams, contribute code they are working on into a shared repository on a frequent and ongoing basis (typically several times a day) for integration into the main code line. An automated build process, often incorporating a degree of unit testing, then runs periodically. If the build is successful, a set of deployable software artefacts is produced and stored for potential release into the downstream environment, e.g. for full integration testing. Not every build is actually released, but the idea is that when downstream activity requires, the latest changes will always be 'ready to go'. Similarly, previous builds are available on demand if they are needed, for example, to support troubleshooting of the live environment.

A number of different types of tool are required to implement CI (Figure 2).

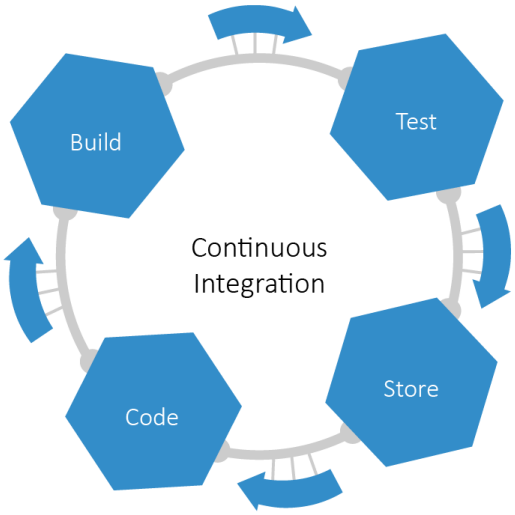

Figure 2
Continuous integration is dependent on a range of tools

Frequently used tools include:

- Jenkins
- Microsoft Visual Studio
- Maven
- Apache Ants
- JFrog
- Nexus
- Artifactory
- Archiva

If it takes weeks or longer to ready and test the required platform needed for deployment, then you have just shifted the bottleneck.

The net effect of this more iterative approach is a reduction in overall cost and overhead within the development process, and more frequent availability of software builds that are fit for taking forwards.

But agile development and continuous integration only get you so far. It's all very well having stable software builds available almost 'on tap', but if it takes weeks or longer to ready the required platform environment needed for full integration testing and ultimately deployment into the live environment, then you have just shifted the bottleneck further downstream. This brings us onto how things have been evolving on the operations side of the house.

Automation of the operational environment

Your operations team is likely to have already implemented at least some level of automation.

Fortunately, for reasons of efficiency, service level management and their own sanity, your operations team is likely to have already implemented at least some level of automation in relation to resource provisioning, configuration and ongoing administration and optimisation cycles. The advent of virtualisation and various forms of cloud computing also means you undoubtedly have mechanisms in place that enable at least a basic level of on-demand allocation of raw resources. The chances are that many, if not most, routine requests for server capacity and storage space can also be fulfilled without the need to go through a hardware procurement cycle.

With this in mind, some of the categories of tooling associated with ops automation may look very familiar to you (Figure 3).

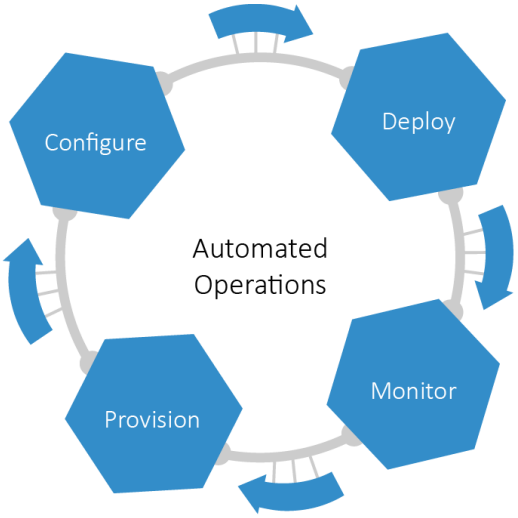

Figure 3
Ops automation utilises its own set of tools

Frequently used tools include:

- Chef
- Puppet
- VMware
- Microsoft Azure
- Amazon Web Services
- Ansible
- Saltstack
- ServiceNow

From a DevOps perspective, advances in ops automation may have started to address the requirements for better integration.

From a DevOps perspective, advances in ops automation may have started to address the requirements for better integration between development and operations. If your organisation has made forays into more advanced virtualisation techniques or private cloud architectures that enable resource pooling and management of 'infrastructure as code', you may have a degree of self-service in place. Within the bounds of operations team comfort, developers and testers may therefore be able to grab many of the resources they need without having to go through a formal request procedure.

Disjoints and excessive scripting

While the natural evolution of activity taking place within many development and operations teams lays a good foundation to support strategic DevOps adoption, if your organisation is like most of its peers, adoption of modern tools and techniques will still be quite patchy and inconsistent. You may, for example, have multiple teams doing agile development, but inconsistently and with varying degrees of discipline and automation. In the meantime, use of traditional methods continues in many areas.

Automation may be used effectively in places, but some administrators may not trust it, or worry about it undermining their worth.

And the same inconsistency is likely to apply in operations. Automation may be used effectively in places, but some administrators may not trust it, or worry about it undermining their worth. They therefore remain wedded to traditional OS-level scripts, or continue to administer systems manually.

In both development and operations, there are then those who get too carried away with the use of their favourite automation tools. Let's take a look at what we mean by this, and some of the early adoption challenges you might come across in general.

Common early automation challenges

Open source software (OSS) is an integral part of the DevOps landscape.

A number of issues frequently arise as IT teams begin to apply automation to more aspects of the delivery process. If left unchecked these can end up impeding higher level progress.

The open source dilemma

Open source software (OSS) is generally considered to be an integral part of DevOps, and for a couple of good reasons. It has led to the rapid emergence of innovative tools to meet the requirements of those leading the automation charge, and has also made those tools freely available. DevOps practitioners can adopt solutions to try new ideas and approaches without going through the usual investment justification and procurement process, or even seeking management permission.

While OSS is key, uncoordinated bottom-up adoption can lead to duplication of effort, redundant solutions, integration disjoints, and lock-in to tooling that's only partially fit for purpose over the longer term.

But this can lead to a dilemma. Should you encourage freedom among staff to use whatever they want to get the job done, i.e. get out of their way completely, or promote a more structured approach?

Allowing freedom can mean teams are never constrained by limitations of the 'company standard' option – a frequent complaint when people are forced to use tools that are a poor fit just because they have already been paid for. When no money is involved, however, it's all too easy to just grab something that appears to be suitable without conducting proper due diligence, e.g. considering how requirements may change down the line or how colleagues have met the same need. Perceived 'coolness' of tools and marketability of associated skills can also influence choices.

This matters because decisions based purely on personal preferences and a parochial view of needs risk duplication of effort, redundant solutions, integration disjoints, and lock-in to tooling that is only partially fit for purpose over the longer term.

If all you have is a hammer…

A frequent problem is overextension of solutions.

Building on the above, there is also a tendency for enthusiasts to sometimes get carried away with the power of their chosen tool, in turn leading to overextension of solutions. With enough scripting it's perfectly possible to use build automation tools to configure hardware, or ops automation tools to run software builds, but neither is necessarily a good idea (Figure 4).

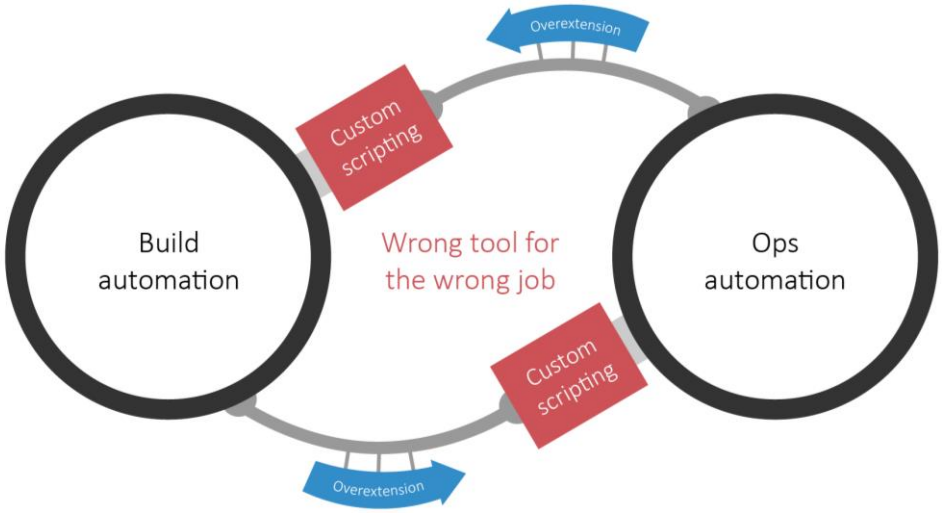

Figure 4

Well-meaning enthusiasts sometimes overextend the reach of tools and create as many problems as they solve

Rapid accumulation of the 'clever' scripts needed when tools are used beyond their natural scope creates undesirable dependencies and leads to an inflexible and fragile overall process.

Such overextension typically comes about when the drive for automation stems from one group in particular. Whether it's the operations team reaching upstream in an attempt to tidy up what they see as a messy and inconsistent software build process, or an agile development team looking to speed up deployment, the intentions are invariably positive. Before long, however, you realise that rapid accumulation of the 'clever' scripts needed when tools are used beyond their natural scope creates undesirable dependencies and leads to an inflexible and fragile overall process.

From a business risk perspective, overextension of tooling also creates an unhealthy reliance on the one or two creative individuals that understand how it all hangs together. If they leave and go elsewhere, they could be difficult or expensive to replace, and figuring out and maintaining what they have put into place could be both difficult and time-consuming.

Time for higher level orchestration?

If you recognise even a few of the challenges we have mentioned, but especially if you are looking around you at a piecemeal landscape held together with scripts and manual handoffs, then it's time to stand back, look at the bigger picture, and investigate ways of orchestrating activity across automation silos. This takes us into the realm of continuous delivery.

The chances are that most automation-related efforts made to date within individual teams can be capitalised on if you put the right kind of overarching framework in place.

From piecemeal automation to continuous delivery orchestration

As you look to implement a more coherent approach to DevOps, the chances are that most automation-related efforts made to date within individual teams can be capitalised on if you put the right kind of overarching framework in place.

In order to understand what we mean by this, let's quickly review the basics of creating a functioning application environment.

Automating environment creation

Whether you are looking to create an environment to support testing, staging or training activity, or the live running of an application, the essential requirements are the same from a deployment process perspective. The overall objective is to end up with an environment in which a stable software build ends up running on an appropriately configured platform layer (Figure 5).

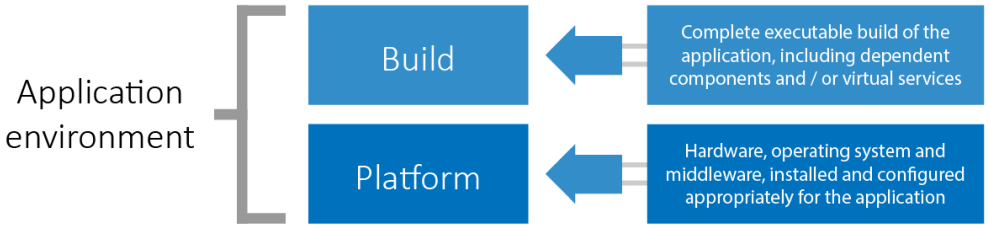

Figure 5

Creating the two key pieces of a fully functioning application environment

Assuming that your development and operations teams have taken steps to optimise activity within their own domains, this desired end state is achieved by bringing together the output from continuous integration and automated ops (Figure 6).

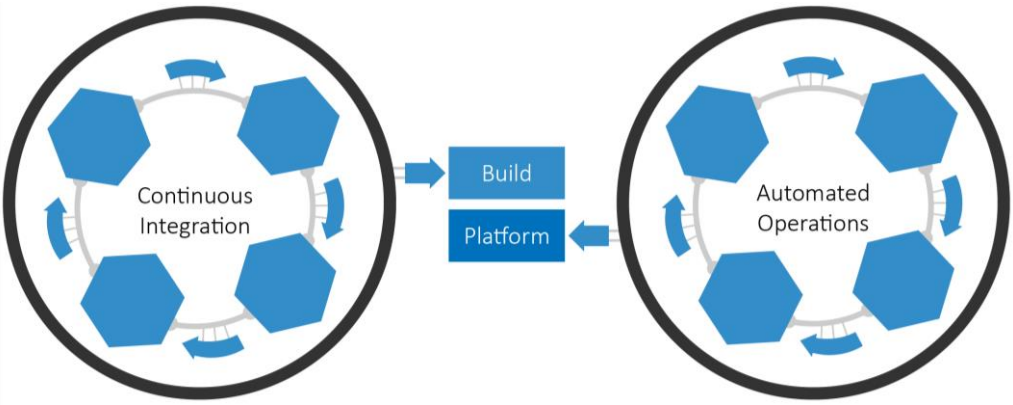

Figure 6

Continuous integration provides the build, and automated operations provide the platform

At this point, the key question is how the two sides of the equation are best coordinated, given that a manual approach is both inefficient and error-prone, and overextension of tools is generally bad news for the reasons we have covered.

The key question is how the two sides of the equation are best coordinated.

Introducing continuous delivery and release automation

The answer to this coordination challenge lies in the notion of continuous delivery, a key aspect of which is smoothing the whole deployment process from end to end. One of the most effective ways of achieving this is through the use of a release automation solution to implement a high level orchestration layer that bridges the gap between continuous integration and automated operations (Figure 7).

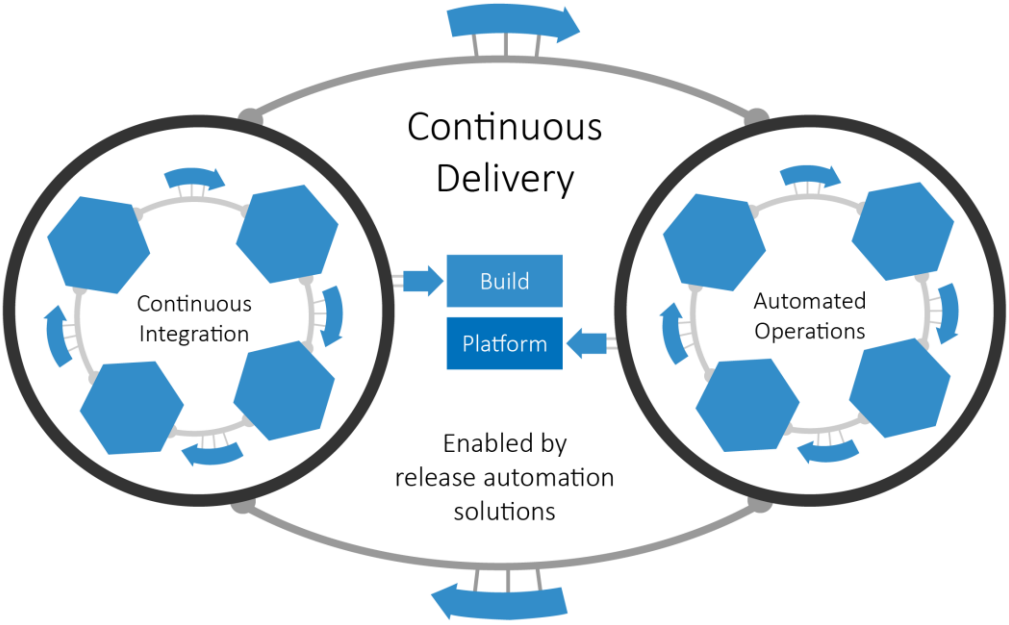

Figure 7

Continuous delivery orchestration enabled by release automation is the tooling pivot for DevOps

When looking at this kind of model, it's important to stress that release automation solutions complement the domain specific functionality already in place rather than trying to replace it. The best way to think of it is in terms of governance and orchestration of the DevOps tool chain. As part of this, the capabilities of specialist automation tools are invoked as 'canned' services within the orchestration process.

Release automation solutions complement the domain specific functionality already in place rather than trying to replace it.

The way in which CA's release automation solution (CA Release Automation) works illustrates this principle very well. For example, a Chef 'recipe' can be incorporated easily into the software deployment process through a few mouse clicks and simple parameter settings, even by a release manager with limited operations skills. And the

level of control is very comprehensive, e.g. in the case of issues arising within the Chef-controlled part of the process, CA Release Automation can stop, restart, wait/fix then restart the run (Figure 8).

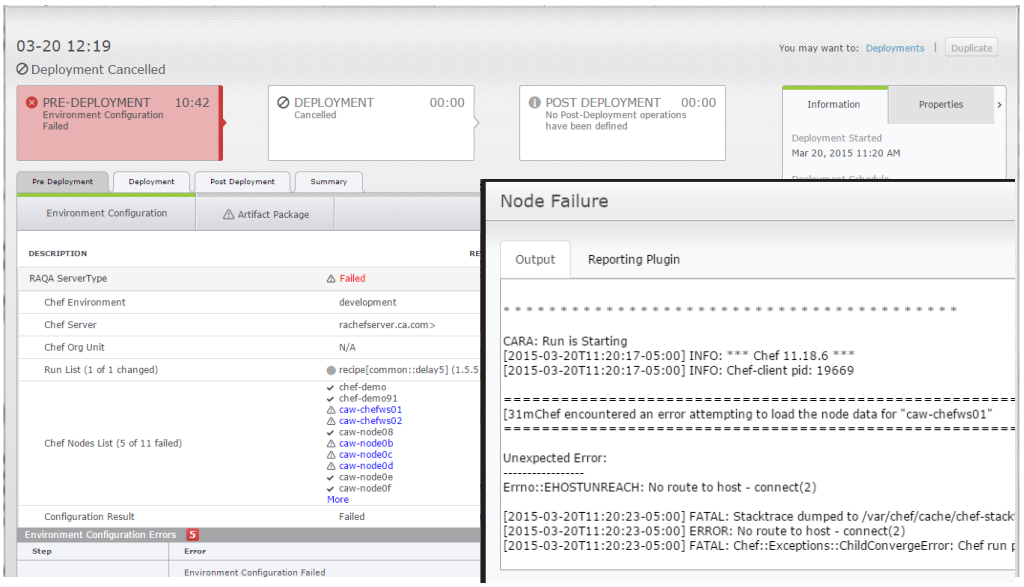

Figure 8

Seamless integration with lower-level automation tools abstracts specialist capability (Chef example)

Through this kind of approach, all of the popular tools and functions involved in the software delivery cycle can be integrated into CA Release Automation, including continuous integration solutions like Jenkins, Maven, Nexus, Artifactory, and other tools used for ops automation such as Puppet. It's even possible to surface resource provisioning services in CA Release Automation from cloud environments like Amazon Web Services or Microsoft Azure. Integrations not supported 'out of the box' can be built yourself or acquired via the user community in the form of custom 'Action Packs'.

All of the popular tools and functions involved in the software delivery cycle can be integrated into CA Release Automation.

One of the benefits of solutions like CA Release Automation is an ability to orchestrate the software release process using highly visual composition tools (Figure 9).

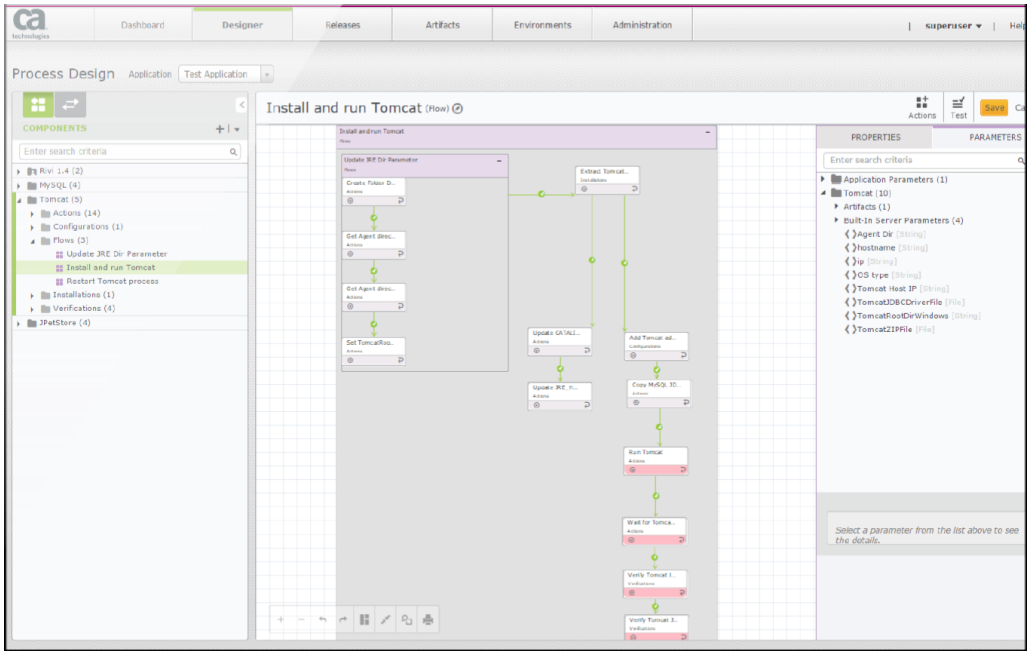

Figure 9

Visual design of the overall process empowers release managers and reduces the dependency on specialist skills

With such capability in place, design, management and automation of the release process is no longer dependent on a select few individuals with specialist skills.

Tools such as CA Release Automation embrace the next levels of complexity and maturity.

Embracing scale and complexity

Orchestrating deployment of a single build of a single application onto a single target platform, while essential for creating individual environments, is only the starting point for enabling serious enterprise class DevOps. Tools such as CA Release Automation embrace the next levels of complexity and maturity.

Orchestrating the deployment pipeline

A good release automation solution will allow you to orchestrate the movement of software builds between environments.

As alluded to previously, for a given application, it is typical in an enterprise IT context to maintain a number of different environments - one used by developers, another by testers, a staging environment to make final preparations for 'go live', and, of course, the ultimate production system. Even this could be a simplification, and for some applications in your organisation it may be relevant to maintain additional environments, e.g. for training or piloting purposes, or to support multiple instances of the application in different divisions or locations.

A good release automation solution will allow you to orchestrate the movement of software builds between all of these environments, i.e. move builds along the deployment pipeline. This is achieved through automated promotion (Figure 10).

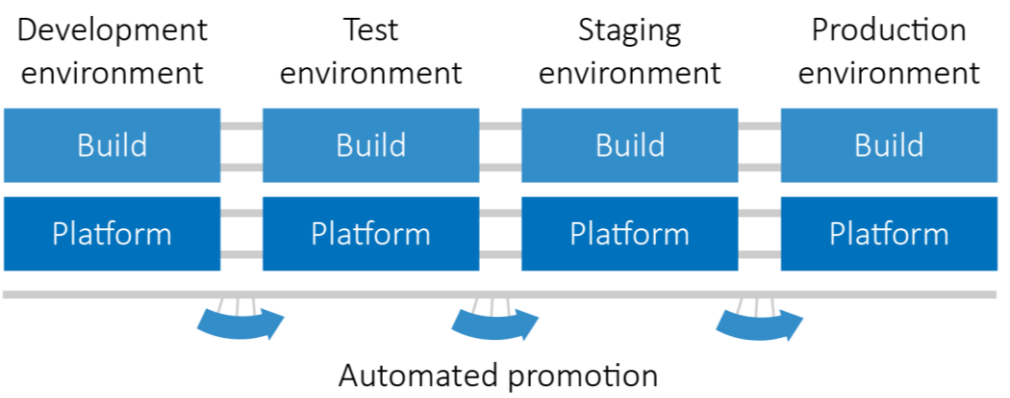

Figure 10

Automated promotion of builds ensures integrity and consistency across environments

In a fast-moving DevOps environment, where continuous integration is producing stable builds maybe once or even several times a day, you are clearly not going to promote all builds along the pipeline. In practice, builds are selected for promotion out of development according to a schedule, based on achievement of a functional milestone, or on demand, e.g. in response to an urgent need for a new feature or fix. Thereafter, they are promoted to subsequent environments as they meet the relevant gating criteria and/or are released by someone exercising manual oversight.

Orchestration components and templates can be created and reused to save time, propagate good practice and generally ensure the consistent implementation of release policy.

To ensure full safety and accountability, for example, final promotion into the production environment will often require a ticket to be run through your service management system. This is why CA Release Automation integrates with both CA and third party service management solutions, the idea being that formal reviews and sign offs can be incorporated at any point in the pipeline.

In practice, you will typically find that many applications follow the same or a similar 'promotion path', depending on their nature. With this in mind, orchestration components and templates can be created and reused to save time, propagate good practice and generally ensure the consistent implementation of release policy.

Some important specifics

If middleware needs upgrading to support a new build, this will be taken care of automatically.

Drilling into some of the mechanics, promotion of a build exploits all of the orchestration capability previously discussed. Chef recipes (or the equivalent), for example, will be invoked at the relevant point to either configure a new platform or make adjustments to an existing target to bring it in line with new requirements. If middleware needs upgrading to support a new build, this will be done automatically.

A particular area of complexity that needs to be handled is the way in which environments necessarily differ along the pipeline. One of the big objectives of automation is to make sure that key aspects of the platform are maintained consistently between environments to ensure predictability, but you can only take this so far. CA Release Automation allows you to synchronise core requirements while catering for environment specific needs.

CA Release Automation allows you to synchronise core requirements while catering for environment specific needs.

In a development environment, for example, installing all of the systems upon which the application is dependent may not be practical. In such cases, a service virtualisation solution such as CA Service Virtualization may be installed into the environment to mimic live system components, e.g. by intelligently simulating web services and other API calls. At the other end of the spectrum, a large-scale production environment is likely to need a lot more resources provisioned and configured than your test system, and even some additional components such as load balancers, high availability (HA) software, and so on.

The release automation solution needs to be able to deal with unexpected situations.

As environments in the real world tend to 'drift' over time, e.g. because of manual interventions to fix or tune systems, the release automation solution needs to be able to deal with unexpected situations, failing with an appropriate alert or fixing in line with pre-defined rules. This requires a good level of handshaking between the orchestration layer (which is primarily concerned with the movement of application packages) and underlying tools such as Chef and Puppet (which track infrastructure and middleware components).

Whatever the outcome of any step in the process, visibility is critical (Figure 11).

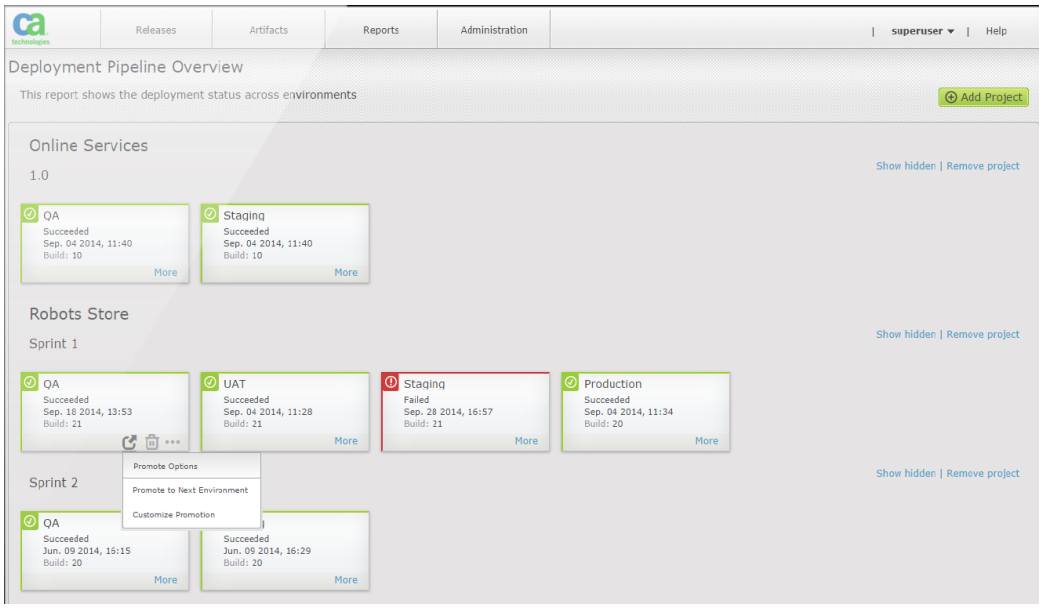

Figure 11

Visibility of activity across the deployment pipeline is critical

But in the real world, you need to be able to manage more than one pipeline.

Scaling out

Implementing DevOps in a large, complex and potentially distributed enterprise setting is a lot easier and more effective if you have a coherent orchestration layer in place.

The broader your DevOps approach, the greater the benefits. As you gain experience, you will want to encompass as large a scope of activity as possible, which in turn means being able to handle tens, hundreds or even thousands of application pipelines simultaneously. That's a lot of activity, a lot of complexity and a lot of accidents waiting to happen if your systems aren't up to the job. And the challenges and risks are accentuated in a distributed environment spanning multiple datacentres, geographies, and outsource partner environments.

While you can theoretically deal with this purely through point-to-point integration of tools, implementing DevOps in a large, complex and potentially distributed enterprise setting is a lot easier and more effective if you have a coherent orchestration layer in place. This is especially the case if the solution supports full cross-team collaboration, governance and component reuse, along with in-built analytics to monitor activity, manage by exception, and gather intelligence to optimise the way your development and operations teams work (Figure 12).

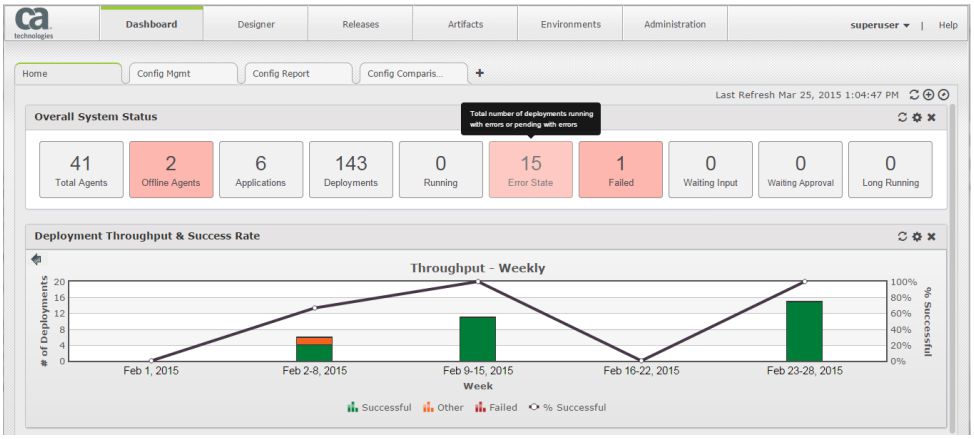

Figure 12

Analytics provides continuous visibility of the release process

Implementing DevOps at scale is not about throwing out all of your open source software and best of breed solutions, and replacing them with integrated suites of 'big iron' commercial software.

Zooming out, as DevOps becomes business critical, it's important to make sure that you have the necessary tooling in place, but what does this mean in practice?

The right tools for the right job

For the avoidance of doubt, it's important to reiterate that implementing DevOps at scale is not about throwing out all of your open source software and best of breed solutions, and replacing them with integrated suites of 'big iron' commercial software. A big part of DevOps is about getting away from the old rigid approach of forcing broad use of solutions that handle a range of requirements, but none particularly well.

It makes sense, for example, to continue to exploit OSS tools in most areas of DevOps. Even if you elect to pay for enterprise distributions and associated maintenance from commercial providers, you will still benefit from rich community and ecosystem support, and advanced availability of features to enable adoption of new ideas and practices. Sacrificing such things for the sake of standardisation is counterproductive.

CA Release Automation represents a strong anchor for any strategic DevOps initiative.

But there are some areas in which consistency allows you to optimise efficiency, quality and service levels without compromising agility, and release automation falls into this category. Indeed tools like CA Release Automation actually enhance freedom through an inclusive approach that takes away much of the drudgery and distraction that prevents developers and operations staff from delivering value. As such, CA Release Automation represents a strong anchor for any strategic DevOps initiative.

Real world experiences

The real test is whether the benefits are being realised in practice.

While the theory of strategic DevOps enabled through a combination of best-of-breed point solutions with an overarching release automation orchestration layer sounds compelling, the real test is whether the benefits are being realised in practice. To provide a flavour of this, Table 1 summarises the essentials of some interesting case studies, more details of which can be found on www.ca.com.

Table 1

Examples of what can be achieved

Business	*Results*
City Index Global leader in Contracts for Difference (CFD), FX and spread betting, transacting in excess of 2 million trades every month for individuals in over 50 countries worldwide.	Through automation of the Value Chain analysis process at the core of the organisation's business, the DevOps team has boosted productivity by reducing the effort to deploy changes by 50%, enabling delivery of software 25% faster.
Tesco Global Fortune 50 retailer and the third largest merchant in the world; operating over 3,700 stores internationally with its expanding online presence, dominating markets as diverse as motor insurance, consumer electronics and clothing.	95% of applications associated with an ambitious international project were deployed into staging within 2 weeks, by which time the manifest and supporting processes had been fully implemented. Deployments were then released reliably and predictably, without the need to modify the deployment process.
Molina Medicaid Solutions (MMS) Subsidiary of Molina Healthcare providing business processing and IT administrative services to Medicaid agencies in five US states and one US territory.	Deployment time for key applications across the business has been consistently reduced from over 12 hours to less than 45 minutes, dramatically decreasing resource requirements, human error, and significantly improving customer service.

Many large enterprises are seeing success from the use of release automation to orchestrate the DevOps tool chain.

These examples illustrate that many large enterprises are seeing success from the use of release automation to orchestrate the DevOps tool chain. Others worth mentioning in the CA customer fold include the large telco, Swisscom, applying CA Release Automation to optimise delivery of software in relation to its IP TV business, and financial services firm ING, using strategic DevOps to support its omni-channel customer engagement activity.

A particularly interesting customer story told at a recent CA Release Automation user group conference was concerned with the concept of 'Continuous Delivery as a Service' (CDaaS). A large financial institution spoke about exploiting reusable components and templates to encourage consistency and the adoption of best practice across its many DevOps teams. When a new project is started, the team is able to immediately anchor it in a continuous delivery framework without having to define policies, workflows and tool chain integrations from first principles.

Continuous delivery and release automation are now very much in use in the mainstream.

Illustrating that DevOps and continuous delivery are not just applicable to enabling the customer-facing part of the business, a large logistics player talked about the benefits in the context of Business to Business (B2B) integration, and development of solutions in the 'Internet of Things' (IoT) arena.

All such stories confirm that continuous delivery and release automation are now very much in use in the mainstream.

Final thoughts

The key to success is striking the right balance.

Implementing DevOps at a strategic level in an enterprise context clearly involves driving a degree of cultural and process transformation. Both of these are much easier to pull off, however, if you make good practices easier to adopt, and remove the sources of friction and distraction that often impede progress. Adopting the right mix of tools and applying them in the right way is an important part of this.

The key to success is striking the right balance between freedom and flexibility on the one hand, and discipline and control on the other.

As you look at how best to achieve this in your organisation, you will undoubtedly conclude that open source software has an important role to play. After all, OSS is woven into DevOps DNA, and it's critical to take advantage of the innovative solutions, vibrant ecosystems and active communities that exist.

As you scale up your DevOps activity, it's important to put the right anchors in place.

In many areas, such as coding tools, build automation solutions and rapidly evolving aspects of IT operations like cloud and containerisation, it even makes sense to encourage experimentation and let practitioners decide for themselves what's best to use to deal with the job at hand. This is all part of staying ahead of the game in a fast moving digital world.

But as you scale up your DevOps activity, it's important to put the right anchors in place to ensure that efficiency, quality and service to the business can be maintained as the levels of activity and complexity increase. One such anchor is release automation software that provides both the control and visibility necessary to orchestrate activity across the DevOps tool chain. In many ways, standardising on a release automation solution that supports an open and inclusive approach enables flexibility to change and evolve in other areas as emerging needs dictate.

Standardising on a release automation solution that supports an open and inclusive approach enables flexibility to change and evolve in other areas as emerging needs dictate.

It also opens the door to bringing traditional applications into the DevOps fold. Back office 'systems of record' and other internally focused software may not have the same absolute need for rapid deployment and refresh, but it can still benefit from the increased efficiency and responsiveness that DevOps offers. Furthermore, avoiding a two-speed approach to IT delivery can help to manage stakeholder expectations, which is important given some of the shifts in attitudes mentioned at the very beginning of our discussion.

With this in mind, the case for strategic adoption of DevOps is pretty clear, and we hope we have helped you to crystallise your thoughts on how best to enable the necessary transformations in your organisation.

About Freeform Dynamics

Freeform Dynamics is an IT industry analyst firm. Through our research and insights, we aim to help busy IT and business professionals get up to speed on the latest technology developments, and make better-informed investment decisions.

For more information, and access to our library of free research, please visit www.freeformdynamics.com.

About CA Technologies

CA Technologies (NASDAQ: CA) creates software that fuels transformation for companies and enables them to seize the opportunities of the application economy. Software is at the heart of every business, in every industry. From planning to development to management and security, CA is working with companies worldwide to change the way we live, transact and communicate.

CA software and solutions help our customers drive enterprise-wide productivity, offer differentiated user experiences and open new growth opportunities.

And, we are able to deliver this value across multiple environments – mobile, private and public cloud, distributed and mainframe.

Our goal is to be recognized by our customers as their critical partner in the new application economy.

To learn more about CA Technologies release automation solutions, please visit http://www.ca.com/gb/devcenter/ca-release-automation.aspx.

Terms of Use