DevOps eGuide

DevOps, that critical intersection between development and operations, has moved into the spotlight of late. While developers and IT professionals have been coexisting for decades, DevOps as a formally recognized practice is relatively new and the inventory of reliable resources exploring how to innervate this movement is still rather sparse. This eGuide provides information you need both to start the DevOps conversation in your organization and to continue the momentum as the industry learns to navigate and strengthen the relationship between the development and IT groups.

In This DevOps eGuide

Overcome DevOps Adoption Barriers to Accelerate Software Delivery

Every team can improve its delivery speed. When teams increase agility and reduce time to market by adopting agile software development practices, DevOps principles, and cloud self-service platforms, they choose a development and operations environment that accelerates software delivery.

Eight Mistakes that Prevent DevOps Success

DevOps is more than just a buzzword; it's about delivering value to customers faster and more reliably. These eight mistakes can dramatically increase cycle time and slow the flow of releases to customers, which is counter to the goals of DevOps.

How Agile Teams Use DevOps for Deployment

Agile practices add value by helping teams detect problems early and avoid repeating them. These practices also help teams get feedback early and often. To extend the feedback loop into deployment, teams are taking a DevOps approach by including the needs of operations teams into the process early.

Harnessing the Power of Collaboration for a DevOps-Driven Organization

From buzzwords to definitions, much has been discussed and debated about DevOps. Yet what it really means is solely up to the IT professionals running the show at thousands of organizations around the world.

Need Another Reason to Consider DevOps?

Automation is a critical component of eliminating waste. A large part of the DevOps movement is centered on automating the infrastructure tasks that take away from delivering value. In addition to the obvious cost savings of automation, it can also breed innovation.

Don't Forget Operations' Point of View in DevOps

Much of what is written about DevOps—a set of principles that helps development and operations teams work more effectively together—is delivered from the perspective of developers. DevOps needs to also take an operations point of view in order to be effective and practical. Let's put the "ops" back into DevOps.

The Buzz on DevOps

Some of the industry's best and brightest share their thoughts on DevOps and its role in agile, QA, and the shift to continuous integration and delivery.

Additional DevOps Resources

Invaluable resources to keep you, your organization, and your practices at the forefront of the DevOps movement.

Overcome DevOps Adoption Barriers to Accelerate Software Delivery

By Chris Haddad

Responsive IT teams overcome software delivery challenges and deliver projects at the speed of business. They work toward simplifying the development and operational experience and shortening project cycle times. DevOps and platform as a service (PaaS) promises to change IT dynamics and create a more responsive IT environment.

Every team can improve its delivery speed. Teams usually face long lead times before project delivery, expensive solution price tags, and uneven policy compliance. DevOps proponents position principles and practices as the solution.

Barriers to DevOps Improvement

Adopting DevOps principles and practices is not easy. Common barriers include overcoming fragmented collaboration practices, discarding outdated IT infrastructure, and replacing previously manual and calcified processes with automated and streamlined pipelines. Without a preintegrated DevOps environment, teams often work for months building an effective tool chain. The DevOps tool chain environment is usually compatible with only a single project template, single run-time environment, or single team. Scaling DevOps requires establishing an environment that can be adopted across multiple project teams and runtimes.

Every team can improve its delivery speed.

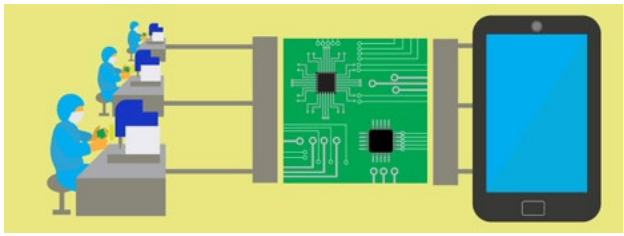
The Path to Responsive IT = DevOps andPaaS

The path to responsive IT requires moving away from traditional application platforms, team structure, and information flows. Responsive IT teams reinvent the application platform and reshape solution delivery. Their architecture intelligently incorporates cloud platforms and DevOps.

Reinvent the application platform

A DevOps-oriented application platform unifies DevOps practices, agile delivery, and PaaS to deliver a consistent, automated, governed, and unified application development lifecycle.

The platform should automate governance and enforce policies. Cloud provisioning should fulfill deployment requirements across all service providers and technologies used by delivery teams. The platform delivers on-demand self-service access

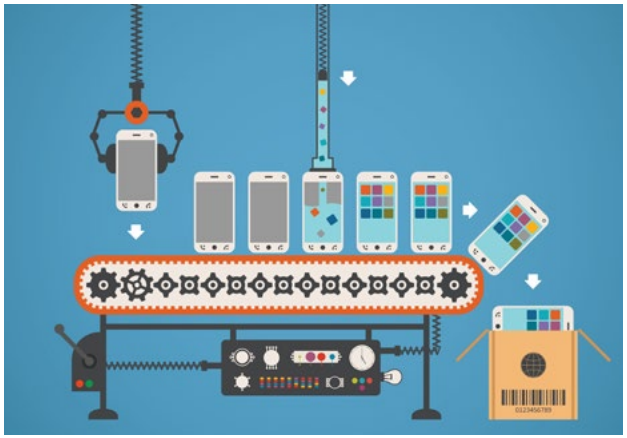


across a standard IT and business service catalogue. The platform should help teams enforce compliance and best practices. For example, governance stakeholders may configure an approval process that includes review gates, approval checklist items, and automated test execution. Because the platform automatically provisions design-time and run-time assets, teams can rapidly deploy projects into a pre-certified run-time environment that ensures use of approved frameworks and services.

Teams reshape development and operations with DevOps

DevOps principles reshape activities to become iterative, incremental, continuous, automated, self-service, and collaborative. Responsive teams adopt DevOps practices of self-service configuration, automated provisioning, continuous build, continuous integration, continuous delivery, automated release management, and incremental testing. Figure 1 illustrates how teams are thinking holistically about the entire application lifecycle, and integrate stand-alone activities into an orchestrated, iterative cycle.

Figure 1. The Full DevOps Lifecycle



Launch Pad

Responsive IT teams rapidly deliver high-quality software. Unfortunately, most IT teams do not have an environment fostering the rapid iteration, streamlined workflow, and effective collaboration required to operate at the speed of now and capture business opportunity.

Disconnected tooling, static environment deployment, and heavyweight governance often impede rapid software cycles, minimize delivery visibility, and prohibit innovative experimentation.

A new, more responsive model is required. When teams increase agility and reduce time to market by adopting agile software development practices, DevOps principles, and cloud self-service platforms, they choose a development and operations environment that accelerates software delivery.

Eight Mistakes that Prevent DevOps Success

By Jonathan Thorpe

DevOps is more than just a buzzword; it's about delivering value to customers faster and more reliably. Over and over again I have witnessed organizations that seem to go out of their way to introduce practices that seem designed to stop value being delivered efficiently. In the ten years that I've spent developing and releasing apps, I've learned that there are critical steps every IT organization should follow that ensure DevOps success.

If you're doing manual tests executed on a build that would never have passed unit tests, you're just wasting valuable time and resources.

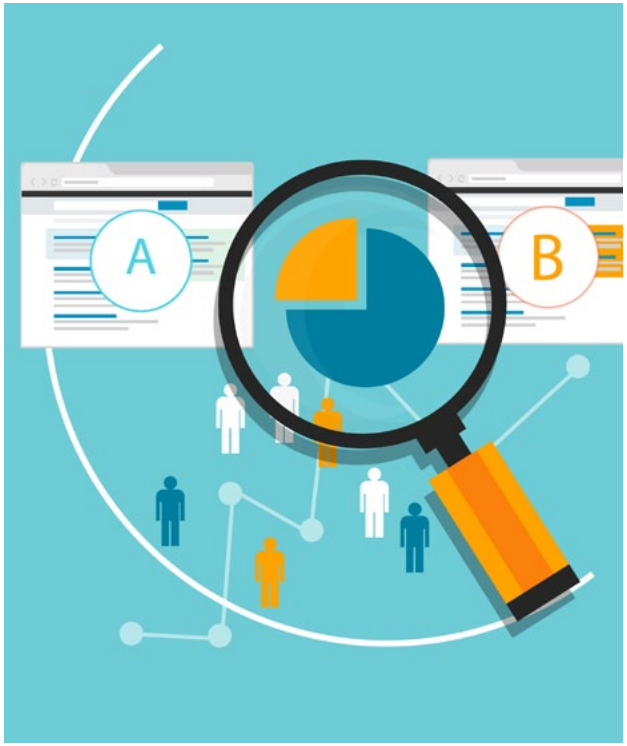
While many of these may seem obvious, experience has taught me that at least one or more of these steps are overlooked. In this article, I will explain eight common mistakes organizations make that can dramatically increase cycle time and slow the flow of releases to customers, which is counter to the goals of DevOps.

1. Not Planning for Scale

Think long term when you embark on the process of bringing development and operations closer together. Create processes that will scale from day one. An example of a process that won't scale is having a single gatekeeper to approve all changes in an automated process. Having a single person able to approve work may work with a few releases per year but that person will become a bottleneck if there are many releases per week. If you're successful, you'll need to scale faster than you ever imagined. Not planning for scale is a fundamental problem I've seen over and over again. Simply put, even if you have amazing success when there isn't as much work going through the system, if your processes don't scale, the processes will quickly become a bottleneck. Allies may quickly become disillusioned and your DevOps initiative may be seen as something that only works for small teams.

2. Not Preparing Your Codebase for Frequent Releases

Another problem I've witnessed, even in projects using the latest technology, involves the codebase. All too often there are a few key files in a codebase that need to be worked on for many, many changes. These can cause significant merge problems, which can result in time being lost on a daily basis. Refactoring is key to moving forward at this stage. Cries of "I can't make this change because it will affect x,y,z," and "It will take several weeks/months..." are strong indicators that a codebase may have problems that can be solved by refactoring. Stakeholders must be involved when the product or feature needs to be refactored. The change involved can take time and isn't without risk, although this risk can be reduced by having unit tests.



3. Not Having Extensive Unit Tests
I've been astounded by how many times I've witnessed teams doing continuous integration (CI) without having unit tests. A CI build only proves that the codebase still compiles. If you don't do unit tests, your automated system tests aren't proving anything. And if you're doing manual tests executed on a build that would never have passed unit tests, you're just wasting valuable time and resources.

4. Not Designing Automated Tests to Scale
A common problem encountered when writing automated tests is that before long, test suites take too long to execute. Even seasoned veterans mistakenly create test cases that are dependent on the outcome of other tests. Tests with dependencies on the outcome of other tests do not scale. Soon, test suites that used to run in minutes will take hours to execute, and as the test coverage goes up, so does the cycle time.

Fortunately, many test cases can be written so that they're not dependent on state from previous tests. While this isn't always possible, tests shouldn't have any interdependencies. When done with virtualization, many tests can be executed in parallel by spinning up multiple environments and executing tests across machines in each environment. This can result in massive time savings and help scale your DevOps solution.

5. Not Managing Your Capacity
Virtualization, including use of the public cloud, can provide enough capacity to service the needs of the development teams, but a common complaint from R&D and QA is that there is never enough capacity. Virtual machines (VMs) are so easy to create that they are built without a second thought. Unfortunately, VMs are frequently not destroyed after use, which quickly eats up capacity and leads to performance problems. Make sure you clean up after yourself. Use automation that allows for VMs to be either destroyed or restored to a clean snapshot and shut down when the VM is no longer in use. A common use case might mean that if QA tests are successful, a VM can be destroyed or reverted back to a clean snapshot and shut down. If the QA tests fail, VMs can be left running or snapshot until there is time for investigation before being destroyed or cleaned up.

Make sure you clean up after yourself. Use automation that allows for VMs to be either destroyed or restored to a clean snapshot and shut down when the VM is no longer in use.

6. Not Using Your Resources Effectively
Spinning up a VM automatically is useful, but what's even more useful is configuring each VM appropriately and putting the latest release onto the system. Use deployment automation tools like Puppet and Chef to ensure that the base VM is appropriately configured. Then use application deployment automation to ensure the latest release is installed on the newly created environment.

7. Not Using One Set of Deployment Processes for All Environments
How can the software your team is building be reliably deployed to all environments? Custom deployment scripts for each environment won't help out with the situation. You should execute the same deployment process over and over again so that by the time the product ends up in production the deployment process will be executed many times.

Model-driven deployment is key to this. The application model should be aware of the environment it is being deployed to. This way, the appropriate values are inserted into the deployment configuration for the environment being deployed to. The end result is that by the time an application is deployed to production, the application deployment model has been tested many, many times.

8. Not Knowing What, Where, When, Who, and How

A CI system such as Jenkins could be used to perform a build, spin up environments, deploy software, and execute tests for each environment. However, a CI system is primarily meant for small-scale builds and deployments.

The problem with enabling this level of automation and self-service is that we are left with many unanswered questions: How do we know what build is deployed to where? What's in the build? What were the test results? There are many other questions that could be raised, mostly related to what can be done with all of the data produced by a large number of builds, deployments, and tests, all potentially being executed in different tools.

Having a higher-level business process model allows for this information to be tracked much more easily to drive future decision making. By adding business process orchestration and having the higher level process execute the CI build, you can start a deployment, run tests, and easily gather and process information about how each build in production made its way to production. Audits become much simpler, and people working in development, QA, and operations will all have a common set of data. From this common set of data, staff at all levels can see what is in a release and how the changes in the release have been managed throughout the entire process.

Automation is the key to a successful DevOps initiative.

Automation is the key to a successful DevOps initiative. It reduces cycle time and moves products to customers faster. But without a good process management and reporting framework, the amount of releases that can be pushed through a system can quickly get out of control. Providing a layer of orchestration that can report on data throughout the entire lifecycle of a release makes it easy to provide metrics that show how your DevOps initiative is having a positive impact on business goals. ■

How Agile Teams Use DevOps for Deployment

Steve Berczuk

Working code is the primary artifact of interest that an agile team creates. However, a successful product delivery means more than building code that can pass developer tests. Remember, code only delivers value once it is deployed. Because developer testing is a prerequisite for successful deployment, it seems that testing deployment environments should be a reasonable thing to do.

Agile practices add value by helping teams detect problems early and avoid repeating them; these practices also help teams get feedback early and often. If you limit your testing to unit and integration tests, you are effectively passing tested code "over the wall" to an operations team to deploy—and this can hide problems. So, the entire team is responsible for deployment.

To extend the feedback loop into deployment, teams are taking a DevOps approach by including the needs of operations teams in the process early. Delivering a simple application to a production environment makes it possible to detect issues early and avoids situations where code relies on mechanisms that won't work in production. While DevOps has challenges, addressing deployment issues early is very valuable.

Extending DevOps to use continuous delivery means that there are opportunities to exercise the deployment mechanism to quickly detect problems, both with the code and with the deployment process. DevOps helps avoid waste by finding problems early.

> ***DevOps helps avoid waste by finding problems early.***

DevOps and continuous delivery help deliver value predictably. The only thing missing is a regular process that validates the deployment environment. Integration and acceptance tests validate that the application works as deployed, but they may not verify that the environment is correct. Though there are a number of frameworks that support developer and integration testing, there aren't many well-defined ways to validate an environment.

Using a tool like Puppet or Chef can help ensure consistency and provide hooks for some validations. These tools are useful for creating known environments, but they don't necessarily tell us if the environment supports the contract we expect. For example, a correctly configured machine could be placed in an environment where an external dependency like a firewall setting has been changed. It's possible to develop tests for all of these things using your own tools, but testing at the DevOps level seems to be missing.

Recently, I had a chance to look at a tool that addresses many of these issues. ScriptRock provides a framework for testing configurations. With a library of existing, configurable tests that you can extend, this seems like a promising tool to identify problems in your deployment environment before they manifest themselves in your application.

Tools like ScriptRock and practices like automated environment testing after configuration changes are great examples of incremental improvements that are central to agile practices. As we test code in the small before committing it to a source code repository, we are pushing automated testing outward and now thinking about how to validate our deployment environments. Not everything will be amenable to automated testing, but automated tests will help us know where to focus our development time effectively.

Much like testing and software configuration management go together, combining testing and environment configuration tools can help you realize agile CM's real value: understanding the impact of a change. ■

Harnessing the Power of Collaboration for a DevOps-Driven Organization

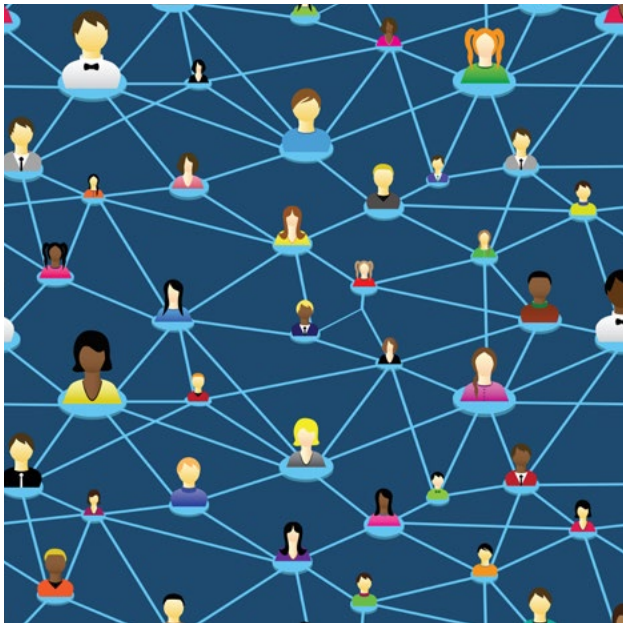
By Cass Bishop

I love tech tools. During my career, I have worked for and consulted with many companies, and every time I begin a project, I immediately look for tools or frameworks to help me complete things faster. For a guy obsessed with new tech tools, now is a great time to be in IT. Git, JIRA, Jenkins, Selenium, Puppet, Chef, Bladelogic, uDeploy, Docker, and Wily (just to name a few) are providing IT with a big-box hardware store full of tools designed to help solve technical problems. These tools are variously pitched, sold, praised, and cursed during DevOps initiatives—primarily because they are good enough for most needs but still leave some critical gaps.

With such an availability of tools, you can check off a lot of the items listed in one of those "X Things You Need for DevOps" blog posts that are published almost daily. Continuous integration . . . check. Automated testing . . . check. Continuous delivery . . . check. Automated configuration management . . . check. Application monitoring . . . check. So, now could you say "DevOps . . . check"? I would argue you will never be able to check that box with the above list of tools because, unless your IT department fits in one room and goes for beers together every Thursday, you are missing the most important concept of DevOps: the need for continuous collaboration about your applications in all of their states, from development to retirement.

> ***...you are missing the most important concept of DevOps: the need for continuous collaboration about your applications in all of their states, from development to retirement.***

Most organizations I have worked with aren't even close to achieving this level of collaboration across development and operations. They are often dispersed across the globe working in different chains of command with different goals, and despite investments in SharePoint sites and using email, instant messaging, and endless conference calls, information is not effectively shared to those who need it. How does a developer in Singapore collaborate more effectively with an operations team in Atlanta? Why can't the incredible number of tools in our arsenal be enough to fix this?



You might say, "We'll give the operations team accounts in JIRA, Jenkins, and Selenium, then give the developer access to Puppet, Wily, Splunk, and the production VMs. They can send each other links and paths to information in each of the different

tools and they can collaborate through email, IM, conference calls, and a wiki or SharePoint site." That sounds OK—until you realize that each of the email threads or chats filled with useful information gets buried in employee Outlook folders or chat logs. When was the last time you heard someone ask to attend yet another conference call or use yet another SharePoint site?

A likely response might be, "We should have them save the chat logs and email threads in the Operations wiki, the Development Confluence site, or that new SharePoint site." With those kinds of approaches, you might be able to find the threads based on string-based searches after a lot of hunting through off-target responses, but more importantly, anyone reading them has no context about how all of the data points in the discussion relate to actual applications, servers, or any other IT asset.

In addition to the lack of context, your IT personnel will spend their days hunting for a needle in an ever-growing haystack of data generated by those amazing tools. Some estimates indicate that as much as 38 percent of knowledge worker time is spent hunting for information to make a decision. [1] Often we give up and make our best guesses based on the information we have at hand. A lot of the time that approach works out fine, but in those cases where it doesn't, it can mean the difference between a smoothly executed project or one that fails and impacts the business.

What if, when your middleware administrator needs to discuss a problem with the UAT messaging engine, she could do so in context with the other experts in your organization? What if her conversation were automatically saved and directly related to the messaging engine, and if the conversation leads to fixing an issue, the lesson learned can be

turned into a knowledge entry specific to messaging engines? Now any IT employee can quickly find this knowledge and see who contributed to it the next time there is a messaging engine issue.

And here's another practical example. What if, when developers want to collaborate with system administrators about higher memory requirements for their applications due to new features, they can pull them into a discussion in each feature's individual activity stream? The admins could be alerted that they have been added to the conversation by their mobile devices and then contribute to the activity stream and even add other participants, such as the operations manager, so he can weigh in on the need for devoting more memory to the correct VMs. This can all be done no matter where the team members are and what they are doing.

That's my vision for what DevOps needs to close the tool chain gap. No one can drop in a DevOps culture for your organization—data federation, visualization, and contextual collaboration are critical to enable that cultural change. Since time immemorial, tools have driven changes in culture. Culture doesn't change on its own or through sheer will of managers. You have to give your teams the tools they need to improve their collaboration. Automation tools and the other types of tools described at the start of this article are important to DevOps, but don't forget the importance of collaboration tools—and don't assume the general-purpose collaboration tools you are using today will cut it, because they won't. ■

...don't assume the general-purpose collaboration tools you are using today will cut it, because they won't.

1. McDermott, Michael. "Knowledge Workers: You can gauge their effectiveness." Leadership Excellence, Vol. 22.10. October 2005, ABI/ Inform Global, p. 15.

Need Another Reason to Consider DevOps?

By Jacob Orshalick

Automation is a critical component of eliminating waste. A large part of the DevOps movement is centered on automating the infrastructure tasks that take away from delivering value. In addition to the obvious cost savings of automation, DevOps can also breed innovation.

With the advent of mobile technology, many products now focus on reducing the time we spend on unnecessary tasks or thought processes. Studies have even shown that the mind adapts to technological resources by only storing how to retrieve information it knows can be immediately accessed elsewhere, such as through a Google search.

A great example is the recent Stick-N-Find product that provides the ability to tag our belongings so that we will never lose them again. A simple Bluetooth sticker and a smartphone app allow you to track down those keys the kids hid in the closet. While I applaud the effort, I find myself losing my smartphone more than my keys.

Freeing the mind of unnecessary daily tasks or thought processes allows more time for creativity and ingenuity. Human history shows this with the age of continued automation leading us to the information age of today. The less time consumed with daily tasks, like washing clothes or ploughing fields, the more time we can spend focused on higher-level thought processes.

Organizations that have focused on DevOps are some of the most innovative in the industry. Facebook, Flickr, Amazon, Etsy, and many more have taken automation to the extreme and have excelled in developing innovative products that are leading the market.

Studies have even shown that the mind adapts to technological resources by only storing how to retrieve information it knows can be immediately accessed elsewhere, such as through a Google search.

I realize that the elimination of waste may not have convinced you to automate that release process that only takes an hour, but just realize that the list of deployment steps your developers have to keep memorized may be hindering their ability to innovate.

If you are looking to get more creativity out of your team, take a look at DevOps. It will not only make your organization leaner but it may spark the innovation you seek. ■

Don't Forget Operations' Point of View in DevOps

By Bob Aiello

Operations professionals are responsible for ensuring that IT services are available without interruption or even degradation in services. IT operations is a tough job and I have worked with many technology professionals who were truly gifted in IT operations with all of its functions and competencies. Many IT operations staff perform essential, albeit repetitive, day-to-day operations tasks that are essential to keep critical systems online and operational. In some organizations, mainframe operators are not as highly skilled as their development counterparts. When developers observe that operations technicians are not highly skilled, they often stop providing technical information because the developers conclude that the operations technicians can't understand the technical details. This dynamic can result in disastrous consequences for the company.

I have also worked with top-notch Unix/Linux gurus in operations who focused on keeping complex systems up and running on a continuous basis. IT operations professionals often embrace the IT Service Management Forum (itSMF) ITIL v3 framework to ensure they are implementing industry best practices for reliable IT services. If you are not already aware of ITIL v3, you probably should be.

The ITIL v3 framework describes a robust set of industry best practices designed to ensure the continuous operation of IT services. The ISACA Cobit and the SEI CMMI are also frameworks that are used by many organizations to improve their process along with both quality and productivity. CM professionals should particularly focus on the guidance in the transition section of the ITIL pocket guide, which describes change, build and release, and configuration management systems (including the configuration management database). With all of this guidance, do not forget to begin with an understanding of the application and systems architecture.

The first thing that I always require is a clear description of the application and systems architecture. This information is not just for my entertainment. For build and release engineers, understanding the architecture is fundamental because all of my build, release, and deployment scripts must be created with an understanding of the architecture involved. In fact, development needs to build applications that are designed for IT operations.

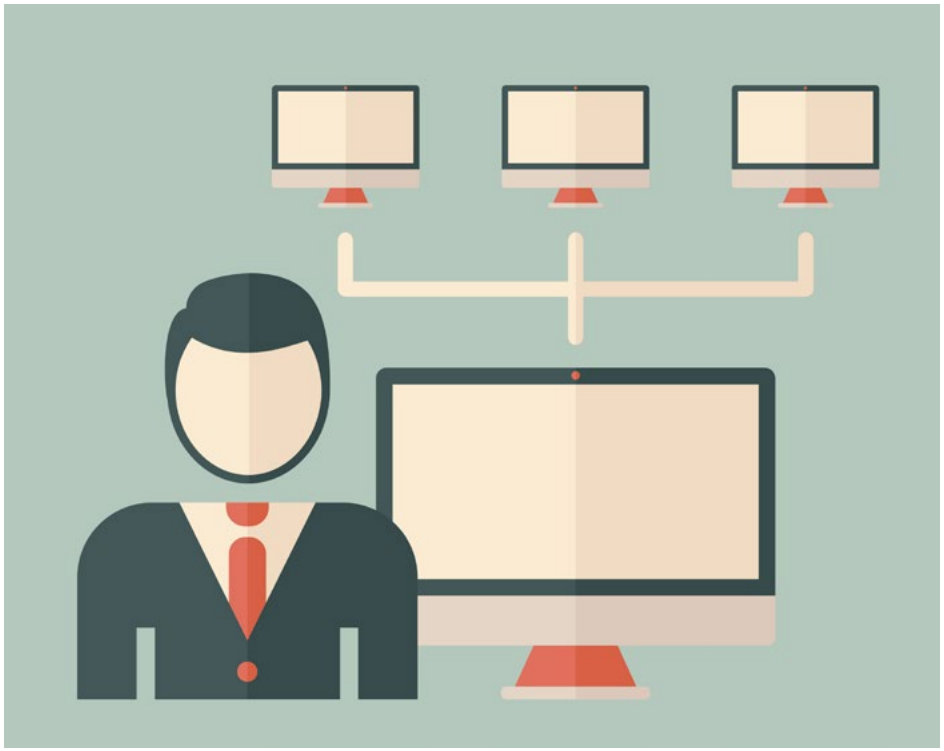
Many developers perform test-driven development (TDD) where code is designed and written to be testable, often beginning with writing the unit test classes even before the application code itself is written. I have run several large-scale automated testing projects in my career, and I have always tried to work with the developers to design the systems to be more easily testable. In some cases this actually included hooks to ensure that the test tools could work without finding too many cosmetic su-

perficial issues, which we usually call false positives. Test-driven development is very effective, and it is my view that applications also need to be designed and written with operations in mind. One reason to design applications with IT operations in mind is to implement IT process automation.

Having effective IT operations is key to any DevOps transformation within an organization.

Effective IT operations teams rely upon tools, including the automated collection of events, alerts, and incident management. When an alert is raised or an incident is reported to the IT service desk, the IT operations team must be able to rely upon IT process automation to facilitate the detection and resolution of the incident. IT process automation must include automated workflows to enable each member of the team to respond in a clear and consistent way. In practice, it is very common for organizations to have one or two essential subject matter experts who are able to troubleshoot almost any production issue. The problem is that these folks don't always work twenty-four hours a day, seven days a week. IT process automation, including workflow automation, enables the operations team to have well-documented and repeatable processes so that IT services are reliable. Getting these procedures right must always start with the application build.

Effective build automation often includes key procedures such as embedding immutable version IDs into configuration items to facilitate the configuration audit. For example, a C#/.net application should have a version identifier embedded into the assembly. You can embed version IDs via an MSBuild script or using Visual Studio and ClickOnce. The Microsoft MSIL Disassembler (Ildasm.exe) can be used to look inside of a .net assembly and display the version ID. You can use similar techniques in Java/C/C++ along with almost every other software development technology. IT operations staff need these techniques to be able to confirm that the correct binary configuration items are in place and that there have not been any unauthorized changes. Builds are important, but continuously deploying code very early in the development lifecycle is also a critical DevOps function that helps IT operations be more effective.



Additionally, application automation is a key competency in any effective DevOps environment. Continuous delivery enables the IT operations team to rehearse and streamline the entire deployment process. If this is done right, then the operations team can support many deployments while still maintaining a high level of service and support. The best practice is to move the application build, package, and deployment process upstream and begin supporting development test environments. These automated procedures are not trivial and it will take some time to get them right. The sooner in the lifecycle you begin this effort, the sooner your procedures will be mature and reliable.

Having effective IT operations is key to any DevOps transformation within an organization. It is all too common for development to miss the importance of partnering with operations to develop procedures that guarantee uninterrupted IT services. If you want to excel, you must include the "ops" in your DevOps endeavors. ■

The Buzz on DevOps

Insight from Around the Industry

On Frequent Releases:
"I think that there's a distinction to be drawn between fear and respect. Releasing a new version of the site or the app is rapidly becoming a routine process, and it's hard to fear something that's routine."

– Simon Stewart

"I see that DevOps has brought a resurgence in the realization that if I've got these practices that significantly flatten the cost of change curve, then I can work in totally different ways."

On Agile's Relationship to DevOps:
"Early on, some of us were worried that "the project managers have taken over the agile movement" and we'd lost all this interesting thinking like from Extreme Programming. And I mean not lost it in individual situations, but lost it as a whole. I see that DevOps has brought a resurgence in the realization that if I've got these practices that significantly flatten the cost of change curve, then I can work in totally different ways."

– Jeff Nielsen

On Continuous Delivery:
"If you had a hiccup because of some configuration issue, then you lost [business]. So continuous delivery becomes really important when you have mobile apps as your main product and you rely on it being high quality and on time."

– Prathap Dendi

On Continuous Integration:
"Continuously integrated builds, particularly at scale, work best when the results of the system have high visibility in your organization. This visibility creates a terrific rallying point."

– Jesse Dowdle

On DevOps Failures:
"The unfortunate thing is that quite often, particularly when you're in a large-scale enterprise with legacy systems and legacy apps, the need to automate feels like the right move, but it can actually really, really hamper your efforts to introduce a DevOps style culture to your organization."

– Mike Baukes

More on Continuous Delivery:
"We have a lot further to go until we have that same kind of mentality that physical goods manufacturers have had for many years, and have had to have for many years. But I think we're getting better at it, and the agile processes have helped a lot, and really focusing on this notion, especially with continuous delivery, we have to always be ready to ship our software. If we're not ready to ship our software today, everyday, then a red light should go off and a horn should sound."

– Anders Wallgren

On the Role of QA in DevOps:
"In DevOps, we are seeing assurance teams elevating to play release automation in conjunction with continuous integration, thereby leading to continuous delivery. This is a classic example of assurance teams bringing in innovative tools and processes to engineer quality outcomes."

– Prasad Mk

"If we're not ready to ship our software today, everyday, then a red light should go off and a horn should sound."

Additional DevOps Resources

More Information for Software Professionals