Liquibase

WHITE PAPER

The DBA's Role in DevOps

The DBA's Role in DevOps

DevOps, a portmanteau of Development and Operations, is a recent IT phenomenon that is beginning to gain mainstream adoption among large-scale enterprises. First developed in web scale startups, the DevOps philosophy is gaining traction with more traditional IT organizations due to its proven ability to streamline delivery processes, reduce inefficiencies and help companies get to market faster. In fact, in a research study[1] conducted by Rackspace in October of 2014, 66% of the respondents reported that DevOps approaches had been implemented in at least a part of their IT organization, and 79% of the remainder indicated that their organizations planned to implement DevOps by the end of 2015.

The top three reasons cited by these organizations for adopting DevOps included the need for more efficiency in the business, a desire to increase customer satisfaction, and to increase application uptime. And these organizations are finding success with their DevOps implementations, with 57% reporting an increase in customer satisfaction. On top of that, 44% also reported that DevOps is helping them get to market faster, and 57% report that implementing DevOps practices has reduced their overall IT spend on infrastructure.

57% of early adopters report that DevOps practices have reduced IT spend on infrastructure

Even though DevOps has worked for these organizations, the philosophy is still relatively new for many IT professionals. And while much has been written about DevOps over the past couple of years from a development or operations perspective, little to nothing has been written with the database team in mind. This is unfortunate, as the adoption of DevOps practices will surely impact the database team as they strive to support increasing release velocities and new ways of working within IT. To lessen that impact and arm database professionals with knowledge about the changes to come, this white paper explores the underlying principles of DevOps, what those mean to the database team, and the different roles that DBAs play in a DevOps implementation.

What is DevOps, Really?

There is wide variability when it comes to understanding what DevOps is, a situation exacerbated by the fact that DevOps is not a prescriptive set of practices with an established doctrine, but rather a philosophy regarding the way that IT works together in order to meet the needs of the business. At its core, DevOps aims to address the needs of the business through affecting existing processes within the IT value delivery stream, the tools and technology used, and the organization's culture.

1 Rackspace (2014). A DevOps Snapshot: Rackspace Survey Shows Business Benefits of DevOps

Through the examination and analysis of existing processes to identify areas that can be made more efficient, such as code handoffs between different departments, DevOps enables the business to get to market faster. Through an emphasis on automating manual, repetitive tasks, DevOps decreases risk in the develop-test-deploy process and increases application uptime. And through its emphasis on breaking down departmental silos, DevOps encourages collaboration between IT professionals and a culture of continuous improvement. Without a prescribed set of practices, though, it's important to note that each DevOps implementation will be somewhat unique and its manifestation will depend on the teams, technologies, and processes which make up your organization.

While the application of DevOps practices will look different depending on the organization, the underlying principles of the philosophy remain the same. Those core beliefs boil down to increasing communication among stakeholders in the IT organization, collaborating together to solve tough technical problems, and automating tasks where possible to increase process efficiency and reduce risk to production environments.

Communication

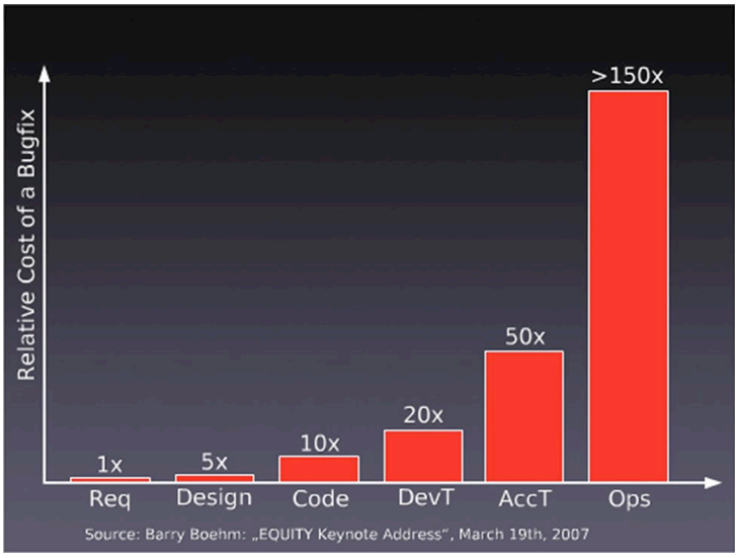
Increasing communication between different stakeholders in IT is the first step in a successful DevOps transformation. By getting different departments or work centers to educate each other on what it is that they do, individuals and teams begin to understand what matters most to their counterparts, and the challenges that come with carrying out those tasks. As an example, it's not typical that development understands how systems are configured in operations, or how those configurations differ from development environments. Perhaps more importantly, without this level of understanding the development team will have little idea of how their changes impact the performance of the application in the production environment or the underlying data platform, setting up the potential for deployment issues later on in the release cycle. The understanding gained through this increased communication engenders empathy between the different stakeholders in IT, increasing the level of caring between different groups in such a way that each tries to prevent creating work challenges for teams located in downstream environments.

Collaboration

As empathy grows in the IT organization, different teams will begin to naturally come together to collaborate on their interdependent processes to remove the inefficiencies and bottlenecks in the value stream. This kind of collaboration is crucial to the long-term success of any DevOps implementation, as it encourages a system-level view of the delivery process. Instead of just trying to optimize the output of their particular work center, IT professionals begin to take a holistic view of the system and collaborate together on ways to optimize the entire value stream.

Part of this collaboration process involves communicating the change plan to stakeholders earlier in the cycle, effectively giving them more time to prepare for an upcoming release, as well as allowing them to provide feedback earlier on. This concept is called 'shifting left,' because it brings production concerns further left in the develop-test-deploy cycle. By making note of specific concerns that will need to be addressed for the push to production, the development team can mitigate potential issues that are sure to cause operations grief. By working on these issues before they become issues, shifting left carries the added benefit of decreasing the cost to the business to fix these issues, in terms of time and resources spent troubleshooting.

At the Equity conference in 2007, Barry Boehm, an IT researcher from the University of Southern California, shared a chart which describes the relative cost of fixing software bugs as a release is promoted through each stage along the path to production[2]. Essentially, what Boehm's research proves is that the earlier an issue is discovered in the SDLC, the less it costs the development team in terms of time to fix the issue, and by extension the less it costs the business which funds the development team's activity.



Automation

Finally, DevOps advocates for the automation of repetitive, manual tasks wherever feasible. The argument for automation is that, once implemented, it brings a level of standardization and repeatability to the release process that will not only make everyone's lives easier, but will also help to improve the performance of the entire application delivery process. For example, by following a very structured, regimented process for updating the database as it migrates along the path to production, your environments will become more in sync over time, eventually eliminating the configuration drift that comes from making out-of-process changes. When your environments are in sync with each other, you benefit from testing environments in Dev and QA that are much closer to what the production environment looks like. This allows you to identify relevant issues much further left in the release cycle, making them cheaper and easier to resolve, as demonstrated by Boehm's research.

It's important to note that the philosophy doesn't advocate for automating every process, but rather only those tasks that are the manual, rote, tedious aspects of your job and consume a great deal of your time. Said another way, automation in DevOps is only a means to an end, the goal of which is to free up time for IT professionals to

2 Boehm, B. (2007). EQUITY Keynote Address. Speech presented at the IEEE International Conference on Exploring

think about more strategic issues, and spend more time collaborating with other groups to drive additional value through innovation in their products and services.

These core principles—communication, collaboration, and automation—form the foundation of a DevOps transformation. When implemented correctly, DevOps creates a virtuous cycle of eliminating inefficiencies through a process of continual improvement. This process accelerates the delivery of applications, and increases the quality of your products and services as realized via increased uptime and the more frequent release of innovative features that will differentiate you from your competitors.

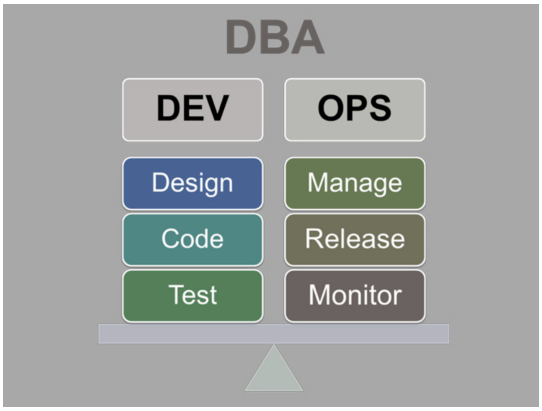
DBA's are uniquely positioned to contribute to the DevOps transformation

The DBA's Role in DevOps

In the existing literature generated from the DevOps movement, there is little to nothing regarding the role that the DBA plays. This is an unfortunate oversight as DBAs are uniquely positioned to contribute to the DevOps transformation, because, as it turns out, DBAs have been practicing DevOps long before it was called DevOps.

As can be seen in the figure below, the responsibilities of the DBA extend through both development and operations, and have for much longer than the DevOps movement has been around. On the development side, DBAs do everything from designing, coding, and testing database objects like stored procedures, packages, functions, and triggers to managing application performance considerations like indexing and partitioning DBAs are constantly contributing to the application through the translation of business requirements into those aspects of the data platform needed to drive features, collect data, and present that data back to users. As data is the lifeblood of the application, and by extension today's digitally enabled business, DBAs fulfill a critical function in DevOps through their stewardship of that data.

On the operations side, DBAs fulfill another critical function in DevOps. Not only do the DBAs design, code, test, and provide infrastructure for the application, they are also responsible for providing the data platform which hosts that infrastructure and managing it to perform as expected. In addition, DBAs have significant responsibilities in coordinating releases, executing releases, and monitoring the data platform after a release to ensure that it continues to perform.



Given that DBAs have been attuned to both development and operational concerns for so long, there is much that they can contribute to the DevOps effort in any organization. To help other groups benefit from their knowledge and experience, there are three roles DBAs should seek to play in a DevOps transformation—the mentor, the guardian, and the hero.

The Mentor

In the spirit of communication, DBAs should look to mentor other groups to further their understanding of how the data platform supports and interacts with the application, and how to properly safeguard that data. To accomplish this, the DBA team should document and publish the database change management process, and provide examples of what complete and effective change requests look like. This serves to set expectations with the development team on what "right" looks like when submitting change requests to the database team. For development teams who author their own SQL scripts, provide examples of clean and effective changes that will help reduce inefficiencies and misunderstandings during handoffs. Lastly, DBAs should seek to educate developers on the things they look for in scripts to ensure that the changes are well formed, safe, and will satisfy the associated business requirement.

In the spirit of collaboration, DBAs should look to maximize their interactions with the development team. If it's possible, attend one or two daily standups a week, or make yourself available for sprint planning or story point estimation meetings. Doing so will ensure that you're constantly aware of the changes development is planning, while providing more time and space to effectively plan for supporting those changes. This interaction will additionally serve to establish the DBA as the subject matter expert on the data platform in the minds of the developers, helping to ensure that they seek out your council before making significant changes.

DBAs should look to increase their interactions with the operations team as well. This, again, will provide opportunities for communication and collaboration that will benefit everyone by reducing some of the friction during handoffs. Find out how operations monitors the database, and provide feedback on things to watch out for to help avoid potential performance issues. Learn about operations' SLAs and maintenance windows to help you in release planning, and seek to understand your organization's release automation practices so that you know how the database component fits into that picture. In the end, understanding what operations' processes look like and the problems that they typically encounter will help chip away at the inefficiencies involved in handoffs between development and operations, benefiting the DBA team in addition to the operations team.

The Guardian

The role of guardian should be no stranger to DBAs, as they have been safeguarding the data since time immemorial. Within a DevOps implementation, though, lies the opportunity for DBAs, again through communication and collaboration, to instill this same mentality in the development and operations teams. Just because DevOps allows IT to move faster doesn't mean that it's acceptable to do so in an unsafe manner, and the guardian role will assist DBAs in making this attitude a permanent aspect of your DevOps culture.

Much like you should document and publish your database change management process to development and operations, you should similarly look to document and publish your organization's best practices and regulatory compliance mandates. This serves to level set the IT organization as to what can and what cannot be deployed into the production database. Examples of items to include might be any naming conventions that you employ to help organize and manage database deployments, harmful practices that won't be accepted, such as setting a default value for a column in a huge table, or the SOX or CCI regulatory database standards your company is subject to.

Next, seek opportunities to automate this aspect of your job by writing tests that parse new SQL scripts and look for violations of your published rules. This can be done using a scripting language like RegExp, and there are many SQL parsing libraries you can choose from in the language of your choice. The goal here is not to replace the validation work that DBAs perform, but to make that task easier by setting up a system that will weed out easily identifiable issues up front, creating more time and space to tackle the difficult ones. If possible, seek to include these code quality checks in your automation framework so that they catch changes as they come out of development. As discussed earlier, this serves to shift these issues left so that you can catch them earlier and get them back to development for re- mediation while the change is still fresh in the developer's mind.

The Hero

Regardless of the safeguards that you put into place, sometimes "bad things happen to good servers." It's during these incidents that DBAs assume the role of hero in their IT organizations, but there are some steps you can take to ensure there is a decrease in the number and severity of these incidents.

Again, in the spirit of communication, DBAs should document and publish their troubleshooting process to benefit each other as well as the broader organization. Formalizing this process provides the opportunity to collect the "tribal knowledge" DBAs have gained over the years on how to properly inspect, discover, and remediate production issues into a central location that is usable by all. Strive to collect this information into a single system of record, whether that's through spreadsheets or an internal Wiki, to assist in troubleshooting and resolving issues faster. In

addition to this tribal knowledge, document the tools that you use in troubleshooting as well as the regular checks you perform during an incident to determine whether or not the fault is originating from the database. This will help in smoothing out the troubleshooting process and improving your Mean Time to Recovery metrics, as well as easing the strain of internal audits.

Next, establish rollback criteria for development and operations. In the event that an incident does occur, it's good insurance to know with certainty that you can rollback the harmful change. To accomplish this, publish the rollback artifacts that are required in order to promote a database change and collaborate with development to ensure that a rollback script accompanies every change. Going a step further, try to provide a means for testing that rollback logic, and if possible, automate those tests.

Lastly, DBAs should remain ever vigilant in their role as hero. This means applying a process of continuous improvement to your troubleshooting process to ensure that it always remains relevant and beneficial. As an example, seek to improve the traceability of your change process over time, which will provide the necessary clues you need to efficiently remediate issues. Additionally, proactively seek and guard against instances of out-of-process change in your database environments, and push to make sure that every database change goes through your formalized, documented change management process.

Conclusion

DevOps, when implemented correctly, provides tangible benefits to both the IT organization as well as the business. But a successful transformation requires a significant degree of change within the organization, and that is always hard to execute. DBAs, with a deep amount of experience straddling the development and operations teams, are uniquely poised to facilitate and influence this change within their organizations for the benefit of all. And in adopting the roles of mentor, guardian, and hero in their DevOps transformations, DBAs have the opportunity to not only streamline their own processes, but to significantly contribute to the business' goals of increasing customer satisfaction and getting to market faster.

Liquibase Enterprise and DevOps

Database change management is a unique challenge when adopting DevOps patterns or an agile development practice. It really straddles two groups: the Developers and the DBAs. The development group is on the hook for producing more and more business critical features and releases at an ever increasing rate. DBAs are tasked with providing a secure, highly available data platform and protecting the integrity of the organization's priceless data. The rate of schema change required by development to satisfy expectations can run headlong into a database

change process that is deliberate and metered by necessity to avoid downtime and data loss. In organizations where these two groups are isolated from each other, you have the makings of a bottleneck in your release process—undermining the promise of DevOps.

In designing and developing Liquibase Enterprise, we performed extensive market validation, holding scores of conversations with professionals throughout the application lifecycle spectrum. In doing so, we learned that the process for managing and updating the database schema which supports an application was at best murky and at worst a black box. Solving the issue of managing database changes was only half of the problem. The other half was to architect the solution in such a way as to bridge the gap between development and DBAs to facilitate collaboration and flexibility between the two traditionally disparate groups, and doing so in such a way as to provide transparency throughout the process for all stakeholders involved.

We found that employing a model-based approach to database change management satisfied all of those requirements. Managing schema change using a model becomes an exercise in incrementally updating a single historical document. Changes are described in a simple, readable format. The application features and business initiatives the changes support are tied to the changes themselves, giving DBAs and developers insight into why a change was made. The reliance on tribal knowledge disappears because everything you need to know about the "why" of a database change is tied to the change itself.

The model-based approach delivers a host of other benefits as well. Instead of writing SQL, the model powers a series of graphical form-based wizards that take the guesswork out of change authoring, facilitating development efforts and the DBA's ability to review those changes before deployment. Prior to execution, the model can be used to simulate proposed changes in memory without touching the database, allowing operations to "rehearse" database deployments before execution, in addition to increasing release confidence when deploying to sensitive environments. Operations personnel can detect configuration drift among database instances more easily than they previously could by eyeballing diagrams or reviewing batches of deployment scripts, and if reality doesn't match the model, the path to remediation is clear.

The database has long been handled with kid gloves, and for good reason. But as our ability to collect and process data becomes greater, the rate at which an enterprise must move on what is learned from that data becomes exponentially greater. Data must be kept safe, but the database must become more agile to accommodate the growing pressure for faster value realization of business initiatives. Companies that acknowledge this and move to adopt DevOps patterns, while including their database teams, will be at a distinct competitive advantage to those who do not.

About Liquibase

Liquibase accelerates database application development and schema change management across the software development life cycle for organizations that practice Agile or DevOps. While there has been a proliferation of application lifecycle management tools to help teams develop and deliver application code faster, innovations for the database have lagged behind, creating a bottleneck in the develop-test-deploy process. Liquibase uniquely solves this problem by managing database changes in lock step with the application code as they are promoted through environments, eliminating the need to track and manage SQL scripts.

A lightweight architecture and tight integrations with release automation tools lead to smooth implementations, increased adoption and accelerated time-to-value for enterprise IT organizations. For the business, Liquibase Enterprise delivers outstanding ROI by increasing velocity, reducing cost and risk across the release cycle, and accelerating time to market.

www.liquibase.com | info@liquibase.com