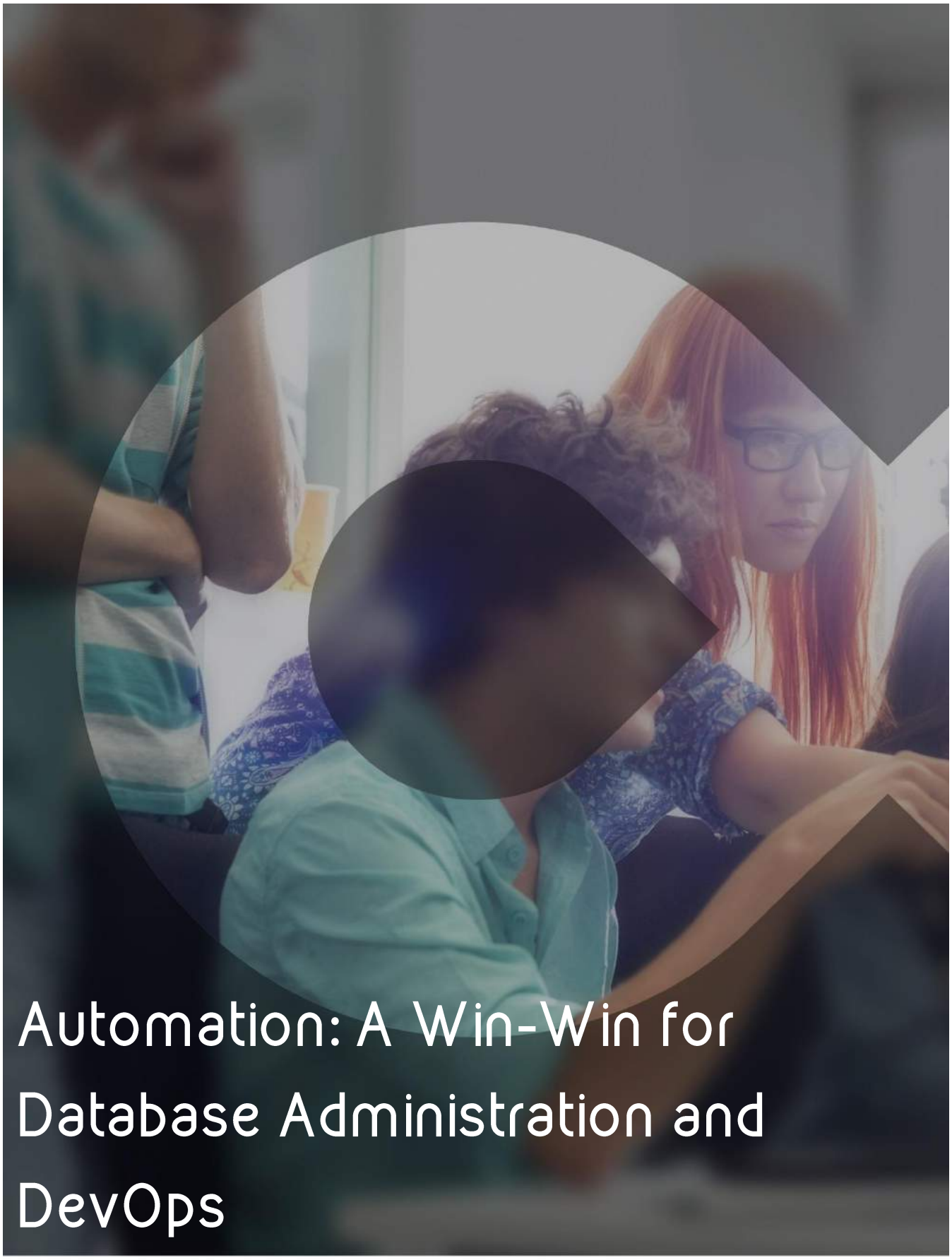


Automation: A Win-Win for Database Administration and DevOps

DBmaestro
DevOps for Database

DBmaestro
DevOps for Database

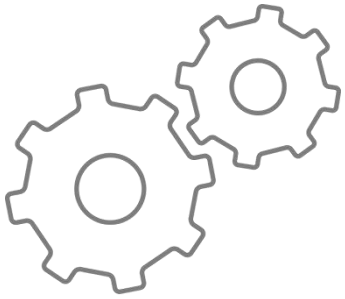
Executive Summary

DevOps transformation is a comprehensive shift in mindset. It represents collaboration between agile development and operations, requiring changes in infrastructure, new tools and expanded skills. As more and more organizations move to a DevOps approach, the speed of their responsiveness to the market and innovation increases. In this shift, it is increasingly important to take into consideration the role of database administrators (DBAs).

They are, in effect, the support staff, coordinators and "traffic cops" for development and for operations. Their impact on the success of DevOps processes or transformation can be significant. Yet, too much of their efforts may be expended in DevOps on trying to align a database, treated like a static system of record, with a dynamic and agile development and production process. The results can be anything from somewhat annoying to catastrophic, costing time, money and resources to recover lost work or handle customer damage control.

This whitepaper suggests that effective DevOps workflow can be ensured, in part, by automating database source control much as is done in code development. As it turns out, automation is a win-win for the DBA and for DevOps.

The DevOps Transformation



"DevOps is about integrating the technical advances of agile development such as continuous integration, automation and version control into an operations context, at the same time as retaining the rigor and discipline of the lights-on mentality." -- Dan North, a pioneer of behavior-driven development (BDD)

A shift to a DevOps approach represents a major cultural change toward a constant two-way flow between development and operations. For operations teams, this means no more code "tossed over the wall" to them with no way to effectively provide feedback to developers. For developers, it means no more flying blind regarding the latest production environment. By breaking down silos and sharing responsibilities, the delivery process is accelerated and more efficient.

Those continuous delivery and agile development principles require a new mindset, new tools and new skills.

In practice, DevOps encourages iterative development in small chunks of code, which are tested and fixed quickly, in the same environment structure as operations. On the operations side, the configuration management code is designed to be repeatedly applicable essentially ad infinitum. And much of these processes are automated – from code testing to workflows to infrastructure.

Among the tools needed are those for building and testing code on a rolling basis (i.e., Jenkins, Bamboo), for source control (GitHub, SVN, Perforce, etc.) for managing, tracking and documenting all changes to the configuration management code (JIRA, ClearQuest, etc.), for configuration management (Chef, Puppet, Salt, etc.), and for measuring environment and application performance (Clarive, etc.).

DBmaestro
DevOps for Database

DevOps Optimizes the Entire Value Stream

Implemented correctly, DevOps is a systematic method of optimizing the entire systems development life cycle. The effects are felt in coding and builds (with source control and continuous integration), quality assurance (with continuous, automated testing), and deployment and delivery (with continuous delivery).

While DevOps emphasizes greater communication and transparency, key to the success of this collaboration is automation. This includes automating the processes for infrastructure changes, workflows, software delivery, and continuous application performance measurement (i.e., merge and build automation, release automation, and more).

A DevOps approach within your organization creates a more coherent production process by breaking down departmental silos and releasing bottlenecks. This is the first step to a more cost-effective use of time in the delivery process, with systematic DevOps automation freeing your personnel to focus on innovation and effectiveness. Development can then be more aligned with your business goals and more responsive to strategic changes.

Adopting a DevOps mentality creates the environment and provides the tools for faster innovation, as well as for the competitive advantage of a faster time to market. When you can deliver better quality software, and you can do it faster, then the ultimate benefits are satisfied customers and increased brand loyalty.

And last, but far from least, product development and operations employees are more satisfied, more engaged and more productive under the DevOps model. Greater cross-team transparency increases the quality and pace of development, while constant feedback increases the sense of accomplishment on a regular basis.

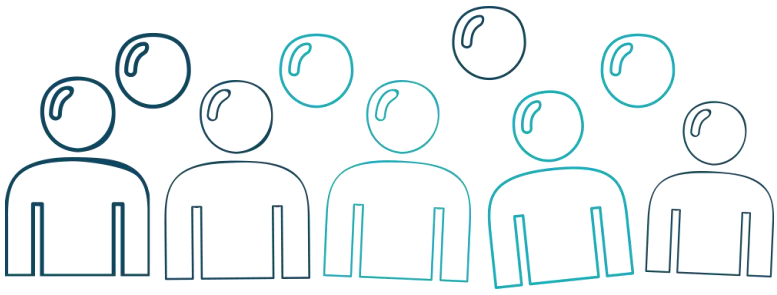
DBmaestro
DevOps for Database

The Database Administrator as Development 'Traffic Cop'

The database, as a core element for software product development, needs to be secure and stable, and perform as expected. Ensuring this is the role of the database administrator, in addition to database planning, development and troubleshooting. He or she works with multiple development teams at once, providing assistance and guidance to the database developers. Then, the DBA must merge code from the different teams, as well as from third-party vendors where needed, and monitor activity in the database. The goal is to make sure there are no overwrites or conflicts between the teams.

In most cases today, the DBA invests a lot of time and effort in manually reviewing code from the developers and preparing the deployment script. At times, this goes beyond fine-tuning and actually involves rewriting entire code segments, simply because the DBA has a better understanding and overview of the database. Similarly, when database problems arise during deployment, DBAs may be called upon to resolve them by fixing unfamiliar code without access to the original developers.

No matter the number of the development teams and their potentially overlapping needs, the DBA is tasked with protecting the integrity of the data and ensuring availability. In order to perform this behind-the-scenes "traffic duty", the DBA must balance the requirements of the various development teams with daily database maintenance routines and administrative responsibilities. This includes defining database management and maintenance processes for the relevant operations team.



With all of that riding on the success of the database administrator, current database deployment methods are surprisingly vulnerable to error, leading to costly rework, downtime, and even data loss. The contributing risk factors are known to every DBA:

- Supporting multiple developers, in increasing numbers of teams and third-party "black box" modules is a challenge. Database configuration drifts are a constant risk due to multiple contributors to the change process, with the quality becoming increasingly unpredictable. And audit trails in these circumstances are often insufficient for regulatory compliance.
- Merging code for deployment means handling dependencies between the different teams, while maintaining enough flexibility to respond to changing deployment plans.
- Dealing with developers and production personnel, as well as acting as arbiter between them at times, is very taxing. Each role requires a different mindset.

- With all of the manual database work required during development and deployment, human error is all but inevitable. And a mistake in the database usually results in cross-application errors, accidental overwrites or downtime in the entire system.

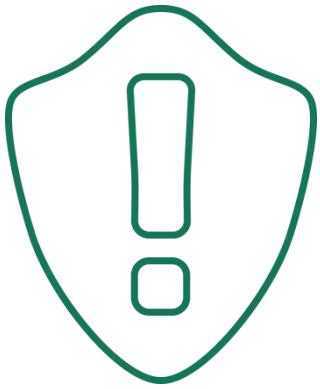
These challenges only become more acute with the current demand for ever-faster time-to-market and consistent customer satisfaction. For strategic business reasons, companies sometimes want multiple features under development at the same time. With all of the development teams making changes to the same database objects, there are ultimately a lot of coupling problems in deployment. This can get even more disruptive, and costly, when a feature has to be rolled back at or after deployment.

Add to this the general customer expectation of minimal-to-zero downtime, and there is clearly great pressure on the DBA just to stay on top of integration and deployments. The problem is that the more attention they need to pay to those aspects of their role, the less time they have for proper maintenance, researching new technologies, assisting database developers, and actual database administration.

How a New Approach to the Database Can Make DevOps a Success (And a DBA's Life Easier)

For database administration to fully contribute to the DevOps workflow, and avoid the pitfalls that can break a deployment, a new approach is needed. The key is a move to greater systematization and automation, ensuring uninterrupted consistency from defining tasks through application deployment.

Specifically, a fully automated enforced version control for the database, much like that used for code development, can address one of the most vexing problems of database stability and security: consistent versioning. This ensures a single source of truth throughout development, creating an automatic process for preventing code overrides and conflicts, configuration drifts, and other causes of instability that can break a deployment.



An effective control mechanism includes a check-in/check-out process and database object locking, preventing employees from straying from a well-defined revision process, whether out of absentmindedness or habit. As no object (table, schema, etc.) can be checked out by more than one user at a time, all revisions are managed and sequential.

In order to mitigate the risk of configuration drift, baseline aware analysis is used to check changed object structure (or content) against the object as it is maintained in the source control repository. That repository preserves a baseline that has been defined for every object and in accordance with every check-in, allowing for three-way comparisons. That is, when a developer wants to commit and deploy his changes, the version control solution should automatically compare the developer's copy, the object baseline (the object as it was when the developer checked

it out), and the most current revision of the object in the target environment (e.g., integration or QA). If a difference between the baseline and the current revision is detected, then that would mean someone else, maybe from another team or project working on another development branch, has already modified the object. In that case, the developer would not be able to deploy his changes, until the conflict between his local copy and the newer updates to the target environments are resolved.

By automatically analyzing code and changes produced by developers, such safe automation can be achieved. The DBA has a clear interest in assuring that bad practice, or non-policy actions, are blocked automatically or trigger a warning, rather than depending on his ability or responsibility to review every bit of code being pushed forward by developers. Practices like dropping an index, rebuilding an index in mid-day, and so on, should trigger a warning before rolling all the way to production.

The result of all this automation is an evolutionary database development that ensures zero downtime, high availability, scalability, and failsafe continuous delivery in any environment.

What Does It Mean for DevOps?

For developers, a reliable and responsive database provides agility and flexibility, while increasing productivity and efficiency, as well as eliminating conflicts with the DBA down the line. For operations teams, it is a promise of security, stability, repeatability, and validity in deployment configurations.

Preventing database-linked errors at the coding or integration phases, as well as raising flags of potential conflicts and non-policy changes, at early stages of SDLC, allows a faster and much more efficient response. Reducing the amount of errors and conflicts early on, long before production, is critical to a better deployment process. The DBA would become less of a bottleneck for the development teams, with less downtime and faster deployment, which translates to better customer experience.

Moreover, many database-linked problems could be resolved by development team leaders themselves, without the need for consultation with the DBA, thanks to database version logs. The overall improved self-sufficiency among the development teams, as a result of the automated database versioning, reduces the number of errors and increases team efficiency. For added security and control in such a situation, the database management solution should be configured to granularly define personnel roles and responsibilities. Such effective separation of duties, alongside enforced version control, is critical to preventing unauthorized or accidental version overrides.

Effective monitoring and versioning should be consistently applied at all stages, with a solution that can analyze database versions and object change histories at any point in their lifecycle. This will provide the audit trail necessary for regulatory compliance and key aspects of quality control. And as technology advances and consumer protection develops accordingly, comprehensive audit trail capabilities will become more and more important for regulatory adherence.

In order to ensure the most streamlined, cost-efficient and documented compliance, the reporting solution should be directly and automatically connected to the database monitoring system. This minimizes manual documentation requirements for development teams and cuts down on the time required to reply to compliance audit requests for information.

In the event of an emergency, it should also be noted, a good source control, safe automation and a monitoring mechanism provide a clear, functional recovery point for the database.

What Does It Mean for DBAs?

Safe automation and source control enforcement in the database reduce the number of errors developers insert into the system, by scanning code for non-policy changes and preventing code overrides. A full and comprehensive object change history provides information on each change, including about the developer who made the change and comments regarding the original business requirement. Labels and a baseline for deployment scripts provide the DBA with additional and valuable information, while risky updates are immediately highlighted or blocked so they can't trickle down to production.

This includes identifying otherwise undocumented hotfixes – the configuration drift problem that is one of the great banes of DevOps and DBAs - implemented to the target schema at any point in development. And if teams are working in separate development environments, only coming together at QA, then identifying changes made to the target environments will also prevent code overwrites.

The source control also identifies who made what change to the database, possibly providing more information on the business requirement behind the change. This is invaluable information for DBAs dealing with problematic scripts during deployment.

Minimizing the errors during development and deployment, while providing additional information about the changes and identifying those that are potentially harmful, allows the DBA to be more efficient. Reducing the amount of "noise", in the form of DevOps errors with which they have to deal, lowers DBA's stress levels a bit and makes the development teams more self-sufficient. This allows the DBA to focus their time on tasks such as tuning, maintenance, planning the deployment process, and providing expert guidance.

DBmaestro
DevOps for Database

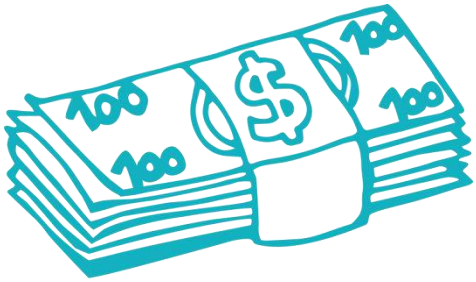
What Does It Mean for Management?

A systematic and automated approach to database management requires fewer resources to run and monitor it – and the results will be far more consistent. By mitigating the risk of human error through automation, the company can influence employees to greater efficiency, prevent work-overs and redo's due to overrides, and track down harmful updates before they are executed.

Database changes and monitoring should also clearly be defined by business requirements, such as tasks from the TFS, Jira and similar collaboration tools. In this way, the database is both up to date and playing its role in meeting business goals.

Safe automation, fully enforced database source control and change tracking, integrated into the development processes, provides compliance with regulatory requirements. This alleviates the need for additional third-party applications.

This end-to-end process, along with open communication and enhanced collaboration internally, optimizes the development to production cycle and reduces its cost. The result is an accelerated time to market and greater customer satisfaction, as the end product is more stable, of better overall quality, and often more quickly reflects user feedback on previous releases.



Taking the Company Forward

Effective database administration can influence the success of DevOps processes, which in turn contributes to the success of the business. The key is an automation that includes safe automation and enforced source control for the database, which prevents many errors long before they get to the deployment stage.

The result is more self-sufficient development teams, faster and earlier corrective measures, as well as more stable deployment configurations. For database administrators, relieved of the pressure of constantly having to juggle and merge various teams' database changes, automation gives them the free time to help their organizations take much larger strides forward in their digital transformation.

Headquarters 300 Baker Avenue, suite 300, Concord, MA 01742, USA +1 978.405.3368
EMEA Headquarters 21 Yagiya Kapaim st. Petach Tikva 4900101, Israel +972.3.9248558

Follow Us: