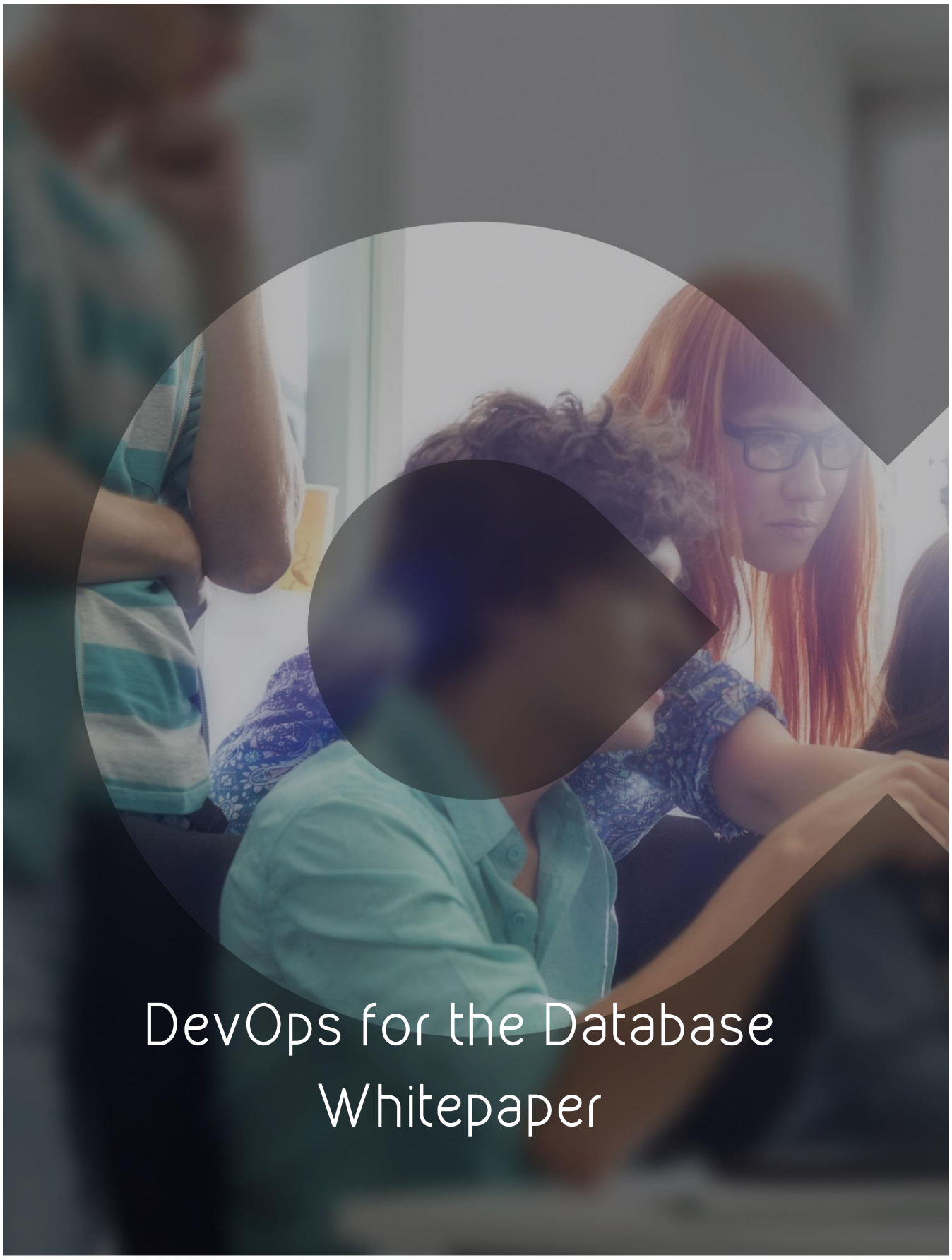


DevOps for the Database Whitepaper

DBmaestro
DevOps for Database

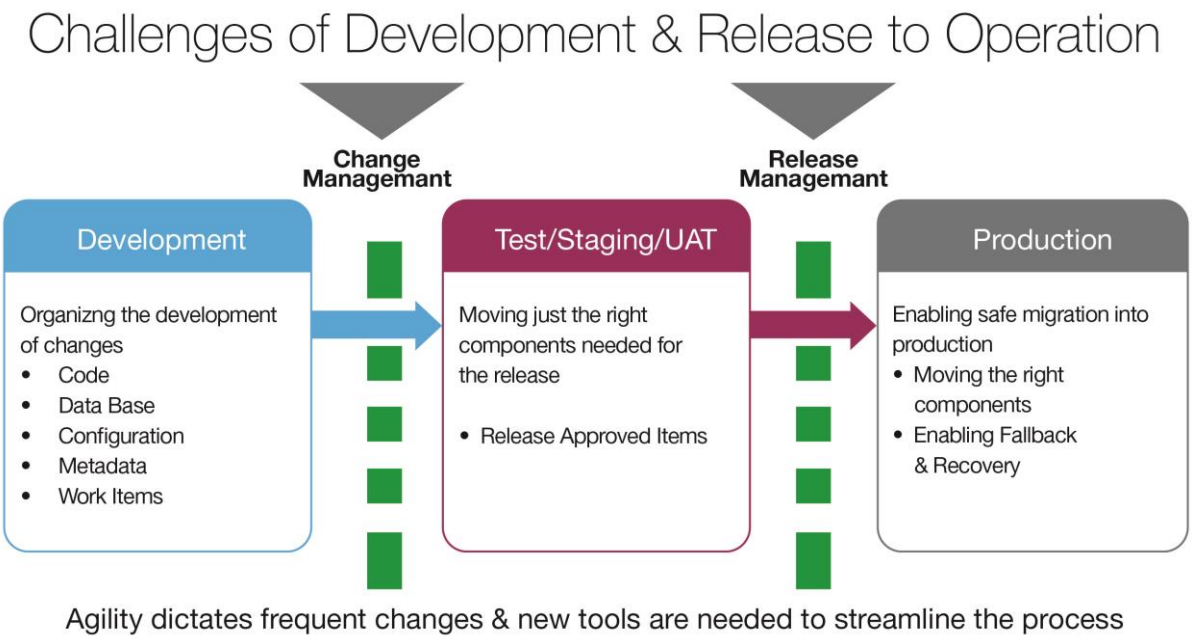
DBmaestro
DevOps for Database

Why DevOps?

Business needs are the most significant driver of change. The ability to do more with less and deliver sooner is what differentiates leading and successful companies from the rest.

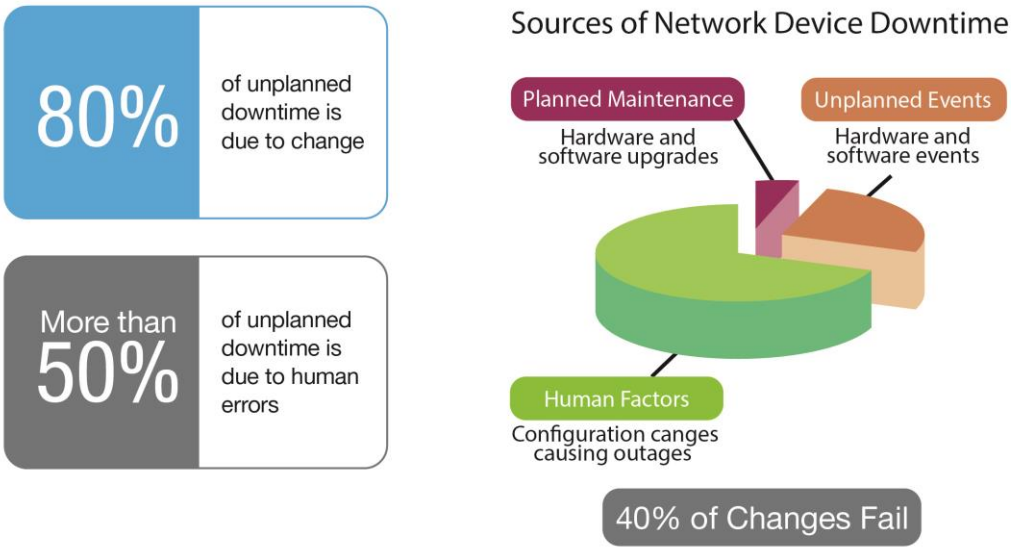
When a competitor delivers relevant features, faster and with higher quality, you're eventually going to lose market share. Investing in sales and marketing campaigns to compensate for your product is expensive and unreliable and you may find that customers are moving to the superior product despite those efforts.

This white paper presents the critical factors that enable a DevOps approach for the Database, a pro-active coordination and collaboration between development and operations teams, to address the tensions and provide for safe, successful, and rapid deployments. Software development has benefited from change management and DevOps methodologies for a long while. Now is the time for databases to benefit as well. You'll learn in this white paper exactly how the tried and tested best practices of DevOps for software can be modified to bring their advantages to database deployments.



Evolution of the Development Cycle

The Waterfall approach to development, which only allowed 2-3 upgrades a year, stopped meeting clients' needs a long time ago. Thus the Agile approach was born. Agile presents a development approach that lets companies move more quickly and deal with ever-changing requirements, all while assuring the highest quality output with fewer resources. Technology companies and IT divisions use the agile development approach, rolling more frequent software and database upgrades, in order to be more responsive to client and market demands. Agile wasn't a revolution; it was a natural evolution responding to changing business needs.

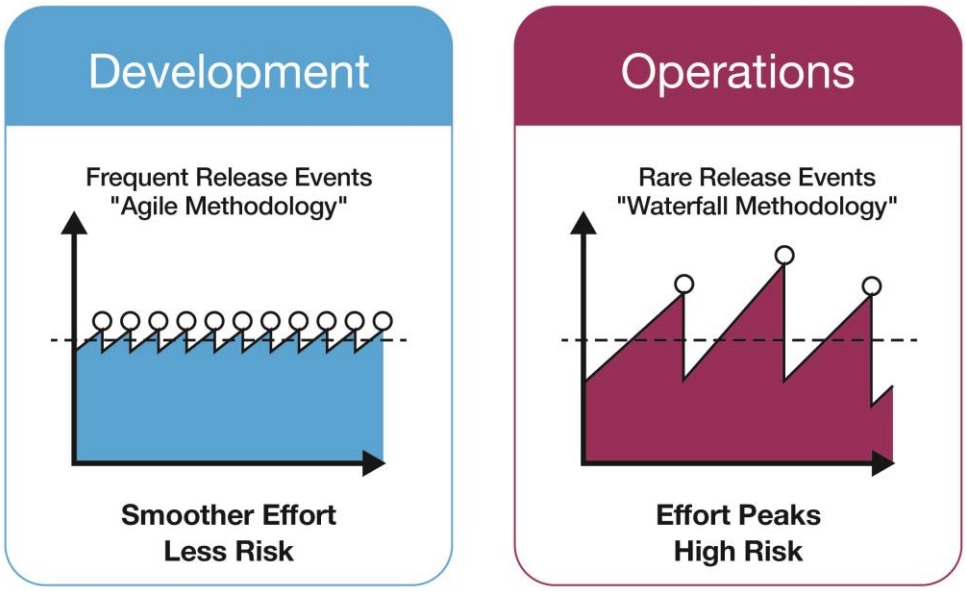


Too Agile?

Yet, as Agile delivered shorter and smoother development cycles, operations struggled to keep up. The short agile production cycles left operations with only two to four weeks between releases to update the production environment and customer sites. Such accelerated schedules only increased the stress and risk of a new software release where multiple teams, handling and considering hundreds of changes to dozen systems and sub-systems, and created the potential of a catastrophic mistake.

The operations chief's interest is to maintain a stable and healthy application conflicts with development's main goal to constantly upgrade applications in response to changing business and customer demands. A way to balance the competing interests of development and operations so they could work more closely together was required: Enter DevOps.

The DevOps approach, as the natural next evolutionary step after Agile, is already successfully used on software development projects. A key source of this success has been DevOps reliance on automation to control the deployment process. However, the differences between the application code and the database code require finding a DevOps automation process that addresses the unique challenges of database deployment. Only then can DevOps truly mitigate the risks associated with high-frequency database updates.



Bridging the Development - Operations Divide

Using the DevOps approach to bridge the Dev/Ops divide is part cultural change and part technological change. The cultural changes required are more effective communication and collaboration between Dev and Ops. However, while fostering a more productive relationship between Dev and Ops will surely contribute to a healthier and more productive organization, it isn't enough to move quality upgrades into production on a timely schedule.

Technology and workflow concepts must support this change, or you will be left with a lot of good will, but not much on the bottom line. You need to increase overall efficiency and reduce the risk inherent in frequent changes and releases.

The DevOps concepts which achieve both increased efficiency and reduced risk are:

1. Adopting Agile over Waterfall: Committing to working in a cycle made up of smaller, more focused and iterative blocks of development, resulting in quicker time to market.

2. Better collaboration and coordination: Using key personnel who ensure everyone across teams and departments are in sync and act as the critical pivot points driving change adoption, and using contemporary collaboration tools to increase both teams' familiarity with the needs, risks and concerns of the other.

3. Automation: Compiling detailed information for each change and using automation-based repeatability will drive safer and less error-prone deployments. The success of frequent changes cannot rely on people's ability to remember everything that's already been done and being able to identify the full scope of what the current change might influence. The key is eliminating as much manual work as possible.

Automation Best Practices

Automation and accessible information are the basic requirements for healthy repeatable, risk-mitigated deployments. Current best practices of software development automation and information collection include:

Change Management
Version control is the most basic means of visibility into what is being changed, and who is driving that change.

Task Based Development
Connecting a development project to the change request or trouble ticket that launched the change. Responding to client feedback with real action is a great plus in serving your market and should be tracked.

Configuration Management and Consistency
Ability to ensure that everyone is working in the same environment; one that resembles the current production environments as closely as possible.

Testing
Testing automation is the best way to ensure that the application is as fully tested as possible. Since development cycles are too short to re-test everything as was done in waterfall projects, automating the smaller testing scope used on Agile sprints contributes to a more stable application.

Reusable Deployment Packages
When you script a task or configure it in a system, it is there forever, fine-tuned and working exactly as you expect it. This allows you to reuse code, avoid manually repeating tasks and opening the door to errors.

Error Handling
Using automation packages that are reliable and safe. A well-written deployment package is aware of its environment and has appropriate mechanisms for error handling, conflict recognition and unexpected occurrences.

Continuous Delivery
Continuous Delivery is a concept in software and database development that calls for a well-defined process, feedback loop, and automation. Continuous processes enable quicker delivery of the most important improvements and fixes, and accelerate delivery of new enhancements.

The ultimate goal of automation is reliable repeatability. Automation is the critical method by which you can ensure changes are propagated through all the necessary phases, from the single developer's sand-box to a team or project integration environment, and then on to testing. Having access to and using the same steps and scripts that have already successfully tested in 'lower' environments means moving into production with less anxiety.

Applying DevOps Automation to Databases

Clearly, deployment automation is critical to practicing effective DevOps. Without deployment automation, releases still require a lot of manual steps and processes. Processes and steps that aren't reliably repeatable, are prone to human errors, and can't be handled consistently with high frequency.

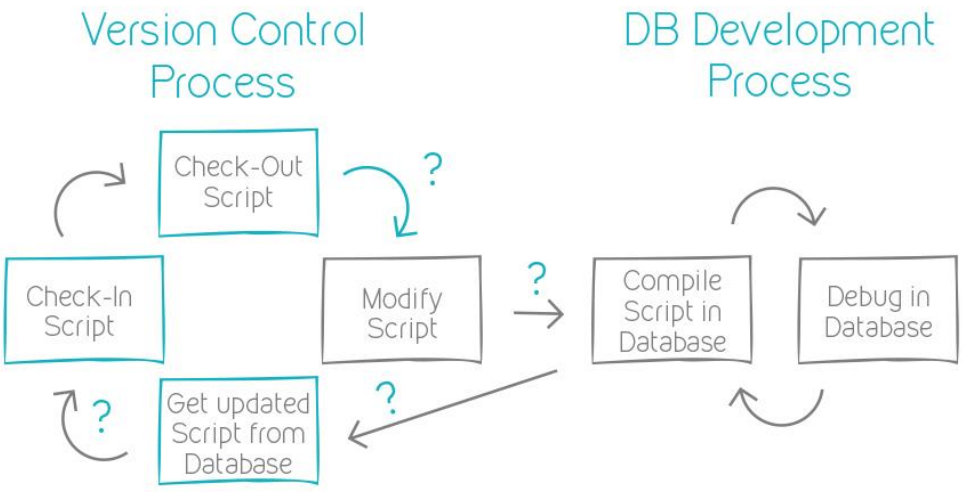
However, the software best practices can't simply be grafted onto database deployments. Unlike software code, a database isn't just a collection of files. You can't simply copy a database among your different development, testing, and production environments. Deploying a database also presents a greater business risk since it holds your most valued asset – your data.

In order to capitalize on DevOps automation advantages for databases and to promote database changes safely and quickly, all the particular components of a database; its tables, functions, and content, need to be protected.

Now, most companies address the particular challenge of frequent database upgrades either by scripting database objects or using the Compare & Sync method. However, both these approaches have shortcomings.

Scripting Database Objects

Using this process, teams script database objects and store these scripts in a traditional software version control system. However, scripts in a version control system aren't connected to the actual database objects they represent, as these two systems are completely separate, and change traceability is dependent on manual non-enforced steps. Out of process chances, code overrides in the database and non-documented changes are not uncommon. Furthermore, the database content, functions, and meta-data aren't usually a part of the version control system in the first place.



The result is that coding and testing of the database code is done disconnected from any real benefits intended by the coding best practices under a true version control system.

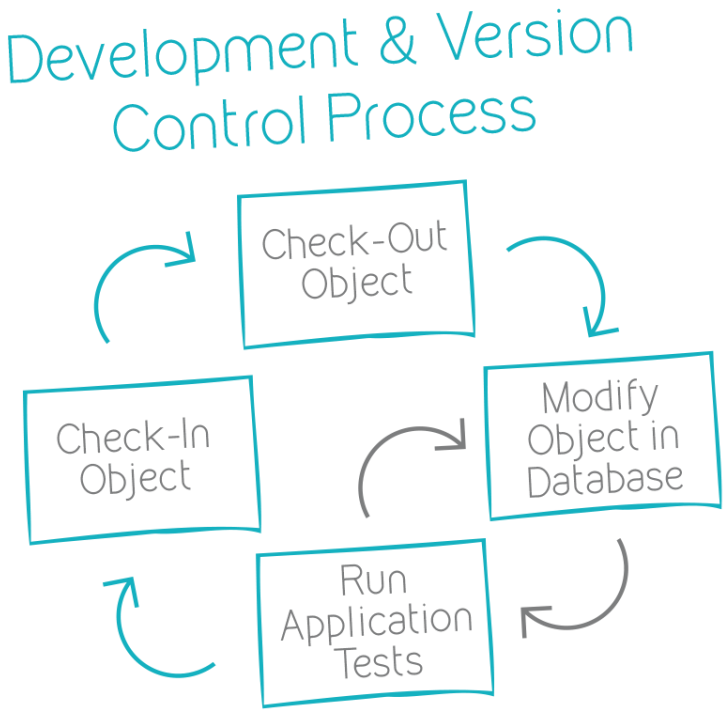
This scripting approach also relies on manually coded scripts. Manual scripting leaves the door open to the risks of missed or faulty dependencies, and updates crashing against changes made in the target environment by another team.

Compare & Sync

An early attempt to automate database upgrades from the last decade involved using tools to create the transition code, known as "Compare & Sync." In this method, a comparison tool examines database objects in a source environment and compares them to the target environment. If a difference is discovered, the tool automatically writes a script to update the target object as a copy of the source object.

However, the compare tool is completely unaware of changes documented in the version control any other approved updates that have occurred in the target environment after development on the first object had begun. If you need to merge code from different teams, assuming the teams are aware of each other's work, you need to merge the code manually.

In other words, the automated "Compare & Sync" method still requires a high degree of manual inspection and script writing, as well as detailed knowledge and communication regarding each change in the deployment process. Otherwise, the predictable mishaps of manual deployments, like overriding current production objects with out-of-date development objects, remain present.



DBmaestro
DevOps for Database

Customizing DevOps/Automation for the Database

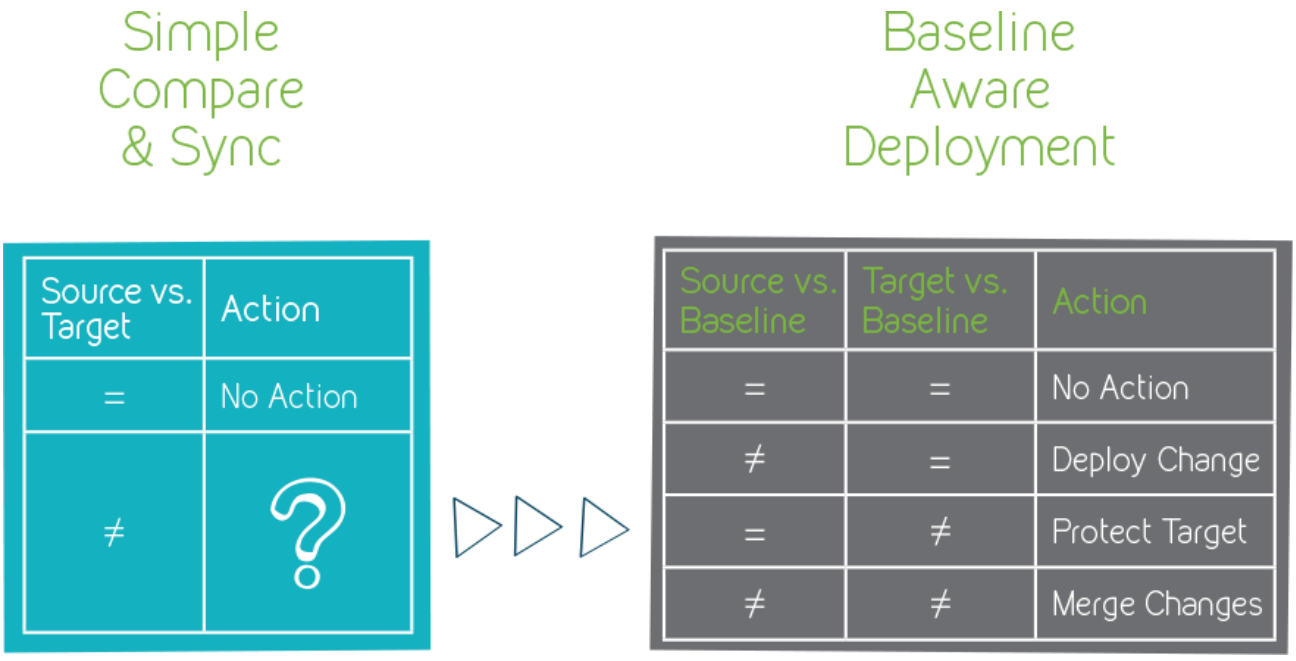
A better solution for automating database deployment had to be found. One that extended the best practices of automated software DevOps change management with adjustments addressing the unique challenges of database updates. To implement effective DevOps automation on databases, an automation solution must:

- Enable proper **database version control** for all types of database objects structure, code, and content.
- **Enforce a single work process** that prevents any out-of-process changes, code overrides, or incomplete updates.
- Leverage proven version control best practices to create a complete audit trail as to who did what, when and why.
- Work harmoniously with task based development, attaching each version control change with a change request or a trouble ticket.
- Ensure configuration management and consistency so all development, testing, and production environments are aligned, or any deviation and difference is well-documented and explained.
- Automate script development to provide reliable deployment scripts with repeatable results every single time.
- Be fully integrated with other critical systems, such as application life cycle, change management / trouble tickets, and release managers

DBmaestro
DevOps for Database

The Most Critical Step - 3-Way Analysis

In addition, any successful DevOps database automation solution must be capable of intelligently dealing with conflicts and merges of code and cross updates from other teams. The solution must be able to ignore wrong code overrides and give the user ultimate control over which changes and merges are pushed out.



In order to achieve these goals, the solution must be able to conduct a three-way analysis on the source (that is, the object after the development work is done), the original (or "baseline") state, and the target (or "Production") environment, identifying the differences among these three states. Team members, gaining full knowledge of the complete scope of the project from the analysis, can review the results and options the solution presents to resolve any discrepancies. The user selects what action, if any, should be taken on the database. The solution then automatically generates the scripts to execute the chosen actions.

In this way, the solution **uses automated three-way analysis and automatic script generation to eliminate both the potential for human error in manual script writing, and the potential for technological error in automated object overrides.**

DBmaestro
DevOps for Database

Reaping the Rewards

Implementing a solution that encompasses all these automation features to meet the particular challenges of database deployments would enable a company to practice a DevOps approach to effective database automation and realize the following benefits:

- **Reduced deployment costs** through streamlining of development process management
- **Minimizing deployment risks** by enforcing change policy and best practices, and configuration management and consistency
- **Improving responsiveness to customers and market needs** with speedier roll-outs of task-based development updates
- **Enhancing cross-team communication** with clear audit trails and transparency regarding work done on objects and the state of different environments

DBmaestro offers a solution that incorporates DevOps automation best practices specifically designed for databases. Join one of our weekly no-strings-attached demos to see for yourself how DBmaestro creates a controlled environment to manage safe database development and deployments.

Pick a date that's most convenient for you and register for our free demo here.

Headquarters 1302 Waugh Drive, suite 290, Houston, TX. 77019, USA +1 281.668.8799
EMEA Headquarters 21 Yagiya Kapaim st. Petach Tikva 4900101, Israel +972.3.9248558

Follow Us: