It’s official: the database is now part of DevOps

The conflict between data protection and DevOps

Don’t just think database DevOps
Think compliant database DevOps
Welcome to the new world of compliant database DevOps

The temptation to include the database in DevOps has been around for a while now, as have the tools that enable it to happen. In a way, it was inevitable. The faster speed of delivery and automation which DevOps encourages streamlines processes, reduces errors, and gets new and better features into the hands of users faster. Exclude the database from the conversation and it acts as a counterweight to the advantages that can otherwise be gained.

Balanced against the appeal of integrating application and database development into the same workflow, however, is the requirement to keep data safe. Particularly at a time when the privacy of personal data and the damage that data breaches can cause is in the news on a weekly, if not daily, basis.

Step in compliant database DevOps. A way of safely including the database in DevOps by matching the desire to deliver software faster with the requirement to protect data and preserve privacy. It gives developers the freedom to keep on developing in the most productive way. It makes sure operations remain in control with review steps included in the process. And perhaps most importantly, it reassures IT management that compliance with legislation is a natural and demonstrable part of the workflow.

Matt Hilbert
Editor
Contents

4  It's official: the database is now part of DevOps

5  The conflict between data protection and DevOps

7  Forget GDPR. Think HIPAA, SOX, SHIELD and the CCPA

10  Questions about HIPAA you were too shy to ask

14  What SOX really means for DBAs

15  Welcome to the new world of curating data, not owning it

17  Don't just think database DevOps, think compliant database DevOps

21  The technical aspects of implementing database DevOps

24  The real advantages of automating your release pipeline

25  How DevOps can also automate the repayment of technical debt

27  Find out where you are on your journey to database DevOps

31  Find a DevOps Partner
You’ll probably have heard of the Accelerate State of DevOps Report from DORA. Now in its fifth year and backed by research involving 30,000+ professionals worldwide, it consistently shows that higher software delivery performance delivers powerful business outcomes. There’s a welcome insight in the 2018 report, however, because it calls out database development as a key technical practice which can drive high performance in DevOps.

Database development has entered the picture because deploying changes to the database is often the bottleneck in software development and slows down releases. To address this, the report investigated which database-related practices help when implementing continuous delivery to improve software delivery performance and availability.

The results revealed that teams which do continuous delivery well use version control for database changes and manage them in the same way as changes to the application. It also showed that integrating database development into software delivery positively contributes to performance, and changes to the database no longer slow processes down or cause problems during deployments.

The starting block is communication, cross-team collaboration and visibility, which echoes Redgate’s own 2018 State of Database DevOps Survey earlier in the year. This showed that 76% of developers are now responsible for both application and database development, and 58% reported their development teams and DBAs work on projects together.

In terms of DevOps itself, the report shows that the highest performing organizations which adopt DevOps release changes 46 times more frequently, have a change failure rate that is 7 times lower, and are able to recover from breaking changes 2,604 times faster.

Crucially, the lead time from committing changes to being able to deploy them is less than one hour in the highest performing organizations – and between one and six months in low performers. Between 46% and 60% of changes deployed by low performers also require some form of hotfix, rollback, or patch.

Beyond those figures, there is one sentence in the report that stands out as the key differentiator and reason for adopting DevOps: “Those that develop and deliver quickly are better able to experiment with ways to increase customer adoption and satisfaction, pivot when necessary, and keep up with compliance and regulatory demands.”

This ties in with Redgate’s ongoing work with helping companies achieve regulatory compliance by adopting database DevOps. The automation it encourages and the audit trails it provides across the database development process ease compliance so that companies can deliver value faster while keeping data safe.

This is just a glimpse of the value the Accelerate State of DevOps Report offers in terms of understanding the drivers that can improve software delivery and operational performance. The main highlights are:

• The improved software delivery and operational performance which DevOps introduces unlocks competitive advantages like increased profitability and productivity
• The key technical practices which drive high performance now include database change management
• Teams which leverage all of cloud computing’s essential characteristics are 23 times more likely to be high performers
• Open source software is 1.75 times more likely to be used by the highest performers
• Outsourcing by function is rarely adopted by high performers and hurts performance
• High performing companies exist in both non-regulated and highly regulated industries

All of which are reason enough to download the full DORA report from https://cloudplatformonline.com/2018-state-of-devops
Data breaches are the new normal – according to the Identity Theft Resource Center there were nearly 1,600 of them in 2017 in the US alone, exposing 179 million records.

Demonstrating the scale of the issue on the other side of the Atlantic, UK retailer Dixons Carphone admitted in June 2018 that it had suffered a hacking attack involving 5.9 million payment cards and 1.2 million personal records.

No wonder companies want to keep data confidential and protected, particularly in a world where they can be fined 4% of their turnover for non-compliance under GDPR rules which protect the personal data of European citizens. And one where data privacy legislation is popping up everywhere.

Sarbanes-Oxley (SOX) and HIPAA are being joined by California’s Consumer Privacy Act on January 1, 2020, and it’s likely to be the first of many new regulations in the States.

At the same time, however, the increasing pace of business means developers need to create and release code in much shorter DevOps timescales. And that includes the database too because Redgate’s 2018 State of Database DevOps Survey revealed that 76% of developers are now responsible for developing databases as well as applications.

This is where the snag arises because, in order to develop and test database code, developers need access to ‘real’ data. Redgate’s survey also showed that 67% of developers use production data in development and test environments, usually in the form of a copy of the production database.

So on the one side you have Database Administrators looking to protect data and account for every record, aiming for anonymity and confidentiality, and on the other you have developers needing up-to-date, production-scale data to properly test changes.

An additional hurdle is the time and disk space eaten up provisioning copies of databases or backups for use in development. This can act as a blocker, either slowing down releases or forcing companies to use out-of-date copies which are no longer representative of the production database.

The need for compliant provisioning

How do you solve this conflict? The database can’t be the bottleneck in DevOps, yet data must be protected.

Organizations therefore need to adopt a four-stage process to keep everyone happy and development and test on track.
1. Provision copies

Duplicating a full database to give a copy to developers is time-consuming and takes up lots of storage space, particularly given the growing size of production databases and the desire of many developers to have their own copy as a sandbox to test changes against.

Solutions like Redgate's SQL Clone have emerged, however, which clone databases and enable copies to be provided to developers which are a fraction of the size of the original database. Importantly, they work just like normal databases and can be connected to and edited using any program, so developers don’t even notice the difference when using them.

Solutions like this smooth out the provisioning process and speed up the DevOps workflow for both DBAs and developers.

2. Protect the data

The faster and more efficient provisioning of copies provides part of the answer, but it doesn’t remove the need to preserve the anonymity and confidentiality of customer data.

That’s where data masking comes in, replacing sensitive data like personally-identifiable information with fictional data that retains the look and feel of the original.

This means the database remains perfectly usable for development and still operates in the same way as the real data for testing and analysis, but the content is secure, helping to meet regulations such as GDPR, HIPAA and SOX.

As well as development and testing, this offers valuable advantages across a wide range of business areas including business analysis and cloud adoption – essentially any situation where there’s a privacy or security risk associated with using real, identifiable data. Companies should look for a solution like Redgate’s Data Masker which enables them to customize the rules which control which data is masked at scale, ensuring company-wide standards and compliance.

3. Automate the process

Different developers and testers will want access to data at different times, probably when DBAs are at their busiest. Therefore, look at self-service tools that automate the process and make it easy for developers to receive copies of the database with information safely masked, when and where they need them. This frees up DBA time to focus on more business-critical activities.

4. Manage access

One of the key requirements of regulations like the GDPR is that companies need to be able to document their processes and provide full audit trails of who had access to which data. Ensure you are able to create a central record of all your database copies, so that you can track this, and look at role-based provisioning to prevent unauthorized access to sensitive information.

Summary

The database needs to evolve faster for DevOps to achieve its full potential – it cannot be the bottleneck. At the same time, companies know the importance of protecting confidential information, and the consequences of failing to do so. Only by adopting a combination of efficient provisioning and data masking will they be able to balance these two demands, remove conflict and speed up development, testing and releases across their business.

This article was originally published on DevOpsOnline in July 2018.
The introduction of the GDPR in Europe caused a lot of companies to start thinking more seriously about data privacy. It also prompted some American companies to reconsider doing business in Europe, and US-based websites like the Los Angeles Times and the New York Daily News have actually blocked access to EU visitors.

Data protection legislation in the USA, however, has been around for a long time, and more is on the way. In most places that companies do business in America, and for most types of business, the changes will move compliance from a minor issue to a big concern. And not just for companies based in the US. Any company with American customers will need to up their compliance game.

That said, what do you need to be aware of right now in the USA, what’s coming up, and what can you do to be better prepared?

Remember HIPAA, SOX and PCI?

Talk about existing American data protection regulations has been drowned out in recent months by the GDPR conversation, but they’re still important and they have far-reaching consequences for many companies.

The Health Insurance Portability and Accountability Act (HIPAA) was introduced in 1996 and provided the first national regulations for the use and disclosure of personal health information (PHI) in the USA. Unsurprisingly, perhaps, the Security Rule of HIPAA requires electronic PHI to be encrypted by “the use of an algorithmic process to transform data into a form in which there is a low probability of assigning meaning without use of a confidential process or key”.

The Sarbanes-Oxley Act, or SOX, came into force in 2002 as a direct response to accounting scandals at companies like Enron and WorldCom to protect investors from the fraudulent activities of corporations. The key to complying with SOX is to safeguard any financial data so that its integrity is assured, and many companies encrypt data regardless of where it resides so that they can guarantee such compliance.

The Payment Card Industry Data Security Standard (often abbreviated simply to PCI), launched in 2004 and amalgamated and aligned the existing standards of Visa, MasterCard, American Express, Discover, and JCB. Given the sensitivity of card data, the masking of card numbers and encryption of other details are common practice to meet the standard.
Have you heard about SHIELD and the CCPA?

In one way or another, many companies in the USA need to ensure they comply with HIPAA, SOX or PCI when collecting, storing and analyzing certain kinds of data. Things are about to get more interesting, however, thanks to two new pieces of legislation which are being introduced at a state level, rather than a national level.

The Stop Hacks and Improve Electronic Data Security Act (SHIELD) is currently set to become law in New York from January 1, 2019. Its aim is to minimize data breaches by introducing stricter requirements for cybersecurity and, importantly, the intent is for it to apply to any company which handles the personal data of a resident of the state. The key take-out is that it will require companies to adopt “reasonable safeguards to protect the security, confidentiality and integrity” of private information.

Similarly, on the West coast, the wide-reaching California Consumer Privacy Act (CCPA) comes into play on January 1, 2020. It requires the protection of broad information like job descriptions, IP addresses and web browsing histories (even when no names are attached), as well as more personal information like names and credit card numbers, and it also applies to any company which does business with any Californian resident.

Why masking and encrypting data needs to be the new standard everywhere

A common strand across every regulation mentioned, whether existing or on the way, is the requirement for there to be measures in place to protect information. That could be personal health data or financial information, credit card details or IP addresses, social security numbers or voice recordings. Access to anything that can be used, by itself or in combination with other data, to identify individuals needs to be controlled.

There's a broader issue here as well, and one which is even more important. Companies doing business in the US are entering an era in which they will need to be compliant with more than one data protection regulation for some of their activities. Expect the smorgasbord of regulations to get bigger too when the residents of other US states start to demand that their privacy is protected as well.

The biggest and best first step companies can take is to bring in measures to protect the privacy of data. Access to production databases needs to be controlled, while backups and copies of databases in other environments like development and testing need to have sensitive data masked.
“Security and risk management leaders responsible for data security and compliance should mitigate the risk of data that enables their organizations’ digital business transformation by adopting data masking and complementary technologies, such as format-preserving encryption and tokenization as a key strategy.”

**Gartner** - July 2018 Market Guide for Data Masking

By making the masking and encrypting of personal data standard practice, rather than an occasional one-off exercise, the worry about whether or not the data privacy requirements of this regulation or that regulation are being followed will disappear.

There are two further benefits as well.

Firstly, data hackers often target environments like development and testing rather than production because security protocols are less stringent. If a data breach does occur, the damage will be limited because the data they’re targeting will be masked.

And secondly, if data masking is the norm rather than the exception, compliance with this element of any data privacy regulation can be demonstrated immediately.

As Gartner recommends in its July 2018 Market Guide for Data Masking:

“Security and risk management leaders responsible for data security and compliance should mitigate the risk of data that enables their organizations’ digital business transformation by adopting data masking and complementary technologies, such as format-preserving encryption and tokenization as a key strategy.”
The Health Insurance Portability and Accountability Act (HIPAA) has been around since 1996. It is designed to protect patient’s confidentiality. Title II (Administration Simplification) which contains the Privacy Rule, Enforcement Rule and the Security Rule, centers around data management, privacy and protection. To be in breach of HIPAA can be expensive and even earn you some jail time (in the worst cases). That said, as a data professional, you will likely have questions about how HIPAA affects you and your day-to-day responsibilities. Rebecca Edwards answers the most common ones that come up.

Why HIPAA?

The Healthcare Insurance Portability and Accountability Act was originally signed into law to “improve the portability and accountability of health insurance coverage” for employees between work. The Privacy and Security Rules were signed in shortly after to protect “any information held by a covered entity which concerns health status, the provision of healthcare, or payment that can be linked to an individual”. Other aims of the HIPAA were to tackle waste, fraud, and abuse of health insurance and healthcare provision.

Does HIPAA apply to organizations outside the United States?

In short - no. HIPAA is applicable to healthcare organizations within the US. Their data is mandated by the requirements of the organization. Even if the people are not US citizens, if they are in a US healthcare system they are also protected. If we consider the reverse, US citizens outside the US, if they are part of a non-US healthcare organization, are not covered by HIPAA.
**What is ‘protected health information’?**

Protected Health Information (or PHI) is any "individually identifiable health information" held or transmitted by a covered entity or business associate. This can be in any form - electronic, paper or even oral. This is information that relates to an individual's past, present or future physical or mental health or condition and the provision of the healthcare to the individual or payments relating to the healthcare of the individual.

HIPAA lists a number of common "identifiers" to make things a bit simpler:

- Names
- Geographic info
- Dates
- Telephone numbers
- Fax numbers
- Vehicle identifiers and serial numbers
- Device identifiers
- Emails
- URLs
- Social security numbers
- IP addresses
- Medical record numbers
- Biometric identifiers (including fingerprints/voice prints)
- Health Plan beneficiary numbers
- Full face photographs
- Account numbers
- Any other unique identifying number, characteristic, code
- Certificate/license numbers

**What constitutes a breach under HIPAA?**

A breach under HIPAA means the acquisition, access, use, or disclosure of PHI in a manner not complying with HIPAA, which compromises the security or privacy of the PHI. This is an extremely broad definition that might make you feel as though even smelling data could land you in trouble. Some examples of HIPAA breaches include: failing to give patients access to their PHI, unprotected storage of PHI (which can lead to laptops or USB sticks being stolen with unsecured PHI*), not logging off your computer/computer system that includes PHI, violation of the "minimum necessary requirement", PHI in an email sent over the internet.

* Unsecured PHI means PHI that is not rendered unusable, unreadable, or indecipherable to unauthorized persons through the use of technology or methodology.

To narrow the scope a bit, HIPAA has specified what's NOT a breach by listing the three exceptions:

1. If an unintentional breach (acquisition, access, or use only) of PHI was made in good faith and within scope of authority and does not result in further use or disclosure.
2. Any inadvertent disclosure by a person who is authorized to access PHI at a covered entity or business associate to another person authorized to access PHI at the same covered entity or business associate, or organized healthcare arrangement in which the covered entity participates, and the information received as a result of such disclosure is not further used or disclosed.
3. A disclosure of PHI where a covered entity or business associate has a good faith belief that an unauthorized person to whom the disclosure was made would not reasonably have been able to retain such information.

The only other exemption for a breach is if it can be demonstrated that there is a low probability that the PHI has been compromised based on a risk assessment to which there are four factors: the likelihood of re-identification/types of identifiers, the unauthorized person to whom the breach was made, whether the PHI was actually acquired or viewed, and to what extent the risk to PHI has been mitigated.
5 What happens if there is a breach/violation?

First thing - stop the breach ASAP. Ensure whatever caused the breach is fixed immediately. Following a breach, covered entities and business associates must provide notification of the breach to the HHS (US Department of Health & Human Services), individuals affected and, in some cases, the media. Notifications must be made without unreasonable delay and no later than 60 days following the discovery of the violation.

The penalties for a breach under HIPAA vary depending on the circumstances of the leak, and the volume of violations. For unknowingly violating HIPAA it is $100 per violation, but in extreme cases for covered entities and individuals who violate under false pretenses, it is a $100,000 fine (up to $1.5 MILLION for repeat violations) and up to 10 years in prison. Fines are issued by the Office for Civil Rights (OCR).

6 Why is data protection suddenly becoming important?

Due to a number of high profile scandals, the public are becoming more and more aware of their rights to protection of their privacy and data. The eventual consequences of those data breaches to the public can in some cases be catastrophic, for example, fraud. As such, support for more stringent legislation has dramatically increased, which is why we have HIPAA and others like SOX, GDPR, SHIELD and the CCPA. The age of technology has made it all too easy to share and discover information, so not only can data be found but also lost and circulated faster than ever before.

7 What rights does HIPAA give to the individual?

Ultimately HIPAA is designed so individuals have easy access to their health information and have more control on the decision regarding their healthcare. Individuals have a legal and enforceable right to see and receive copies upon request of the information in their medical and other health records maintained by their healthcare provider and health plans.

Patients are also able to designate a personal representative (who might already have authority to make healthcare decisions for the individual) who also then has the right to access PHI.
8
Does HIPAA affect the way we do development work?

The Privacy Rule addresses how patient information can be used and disclosed. In the Minimum Necessary Requirement, it states that covered entities are required to evaluate their practices and enhance safeguards as needed to limit unnecessary or inappropriate access to and disclosure of protected health information. For development we can assume this means that it is no longer appropriate to be working with real data. Data retention is similarly a key part of HIPAA - individuals have the right to access information at any time.

Whilst there is no HIPAA medical records retention period outlined, there is a requirement for other HIPAA-related documents (such as but not limited to; logs recording access to and updating of PHI, authorizations for the disclosure of PHI, IT Security system reviews) to be kept for a minimum of six years from when the document was created or when it was last in effect. This is outlined in CFR §164.316(b)(1).

9
What are the risks I should consider?

Where is the data being stored, received, maintained or transmitted? Who has access to it? Is it controlled? These questions might seem obvious, but data is the biggest risk to your compliance. Organizations need to be very clear where, why, and how PHI is stored, who has access, and what exactly happens to this data. It’s important to keep an audit trail of all activity around the records to be able to prove your compliance. It is also worth identifying and addressing potential threats to your PHI. Become proactive rather than reactive in those vulnerabilities. Consider your network security, training members of staff and reducing access points to PHI internally.

10
Is HIPAA JUST about health data? What about non-medical PII data?

PHI is any “individually identifiable health information” held or transmitted by a covered entity or business associate. You can refer to the answer in question 3 for more details about what PHI is. The identifiers listed in question 3 include data examples that are not specifically health related (example: names, emails, telephone numbers, etc), however when combined with health information about that person, make such information PHI. Therefore, non-medical PII data still needs to be protected under HIPAA.

Please note: HIPAA is a complex piece of legal legislation. Your organization is responsible for understanding the full requirements. This article summarizes some of the details, but further research is always recommended when ensuring full compliance.
What SOX really means for DBAs

Unfortunately, this is no longer the case. With legislation like the GDPR, PCI, CCPA, SOX and HIPAA, the requirements for protecting and preserving the integrity of data are more critical than ever, and part of that responsibility falls with you, the DBA.

Introduced in 2002, SOX is a US federal law created in response to several high-profile corporate accounting scandals (Enron and WorldCom, to name a few). The public and shareholders alike were in an uproar about the fraudulent activities that came to light and companies everywhere were subsequently expected to raise standards to address their concerns. Corporations needed to change the way they worked from the top down and, by law, that means the DBA too.

So what does that mean for a DBA?
For DBAs this meant introducing new procedures around protecting data, creating backup and recovery processes, and ensuring the auditing, encryption and restricted access of regulated data. Importantly, SOX isn’t like many of the other regulations where protecting personally identifiable information (PII) is the main goal. Instead, financial data is the primary focus when trying to maintain compliance.

This information will likely need to be made public sooner or later, so data breaches are not the biggest problem. You need to ensure that data doesn’t get inserted, updated or deleted without being recorded or, worse still, without your knowledge. It’s about ensuring shareholders have a transparent view into the company.

These prerequisites might sound simple but in practice they can be quite difficult to meet. Keeping track of who changed what, when, where, and how across all activities can seem daunting. There is also the concern that it will have an impact on performance and disk space.

One approach may appear to be reducing who has access to the data. If we reduce the number of people who have access, after all, we reduce the work needed to track those interactions.

A development team might protest, however, that it is absolutely critical they have access to real data in order to accurately test their changes and work effectively. And they’re correct. Generated data as a solution does not tend to test well and can slow down your release processes. Striking a balance between data protection and data performance is the challenge.

Compliance with performance
The worry that processes in place to ensure compliance can bottleneck the development process is a very real one. So how can you have the best of both worlds?

The solution is to provision and mask copies of production databases with realistic, workable data, and this idea of masking and provisioning going hand in hand is called out in the recent Data Masking report from Gartner:

“Test data (or copy data) virtualization is a technology that is increasingly popular, when used in combination with SDM (Static Data Masking), to speed up the provisioning of and updates to target environments, in addition to significantly reducing the amount of storage required by these environments.”

So while compliance does have an impact on database development, it doesn’t mean it has to slow it down.
Faced with the new challenges of data protection – and increased levels of oversight – many of us working with SQL Server have come to the same conclusion. Quite simply, the column names and free text descriptions available to us to describe the data in our care (or data-containing artefacts like columns, tables and databases) are no longer sufficient.

We need to more accurately classify, map, tag or catalog data in order to support automated behaviors, simplify decisions, and – most importantly – tell others how we are handling precious assets. They can then be confident we’re doing the right things, and our actions are aligned with their priorities.

But how? What is the simplest slice of tagging, etc, that we can apply to data to achieve the desired outcome? Is it a label or tag on a column? Or is it a catalog of information that can be tied to that column, wherever it may be?

Museums and art galleries have been using both catalogs and labels for some time and there are numerous guides that show how they have solved common problems around record keeping and context, some of them quite charming.

As I read the simplest guides I could come across, I felt the parallels with the challenges faced by data professionals were ever more striking. So much so that in this new era of regulatory scrutiny, we would be wise to think of ourselves as the curators of data rather than the owners. We would be well advised to develop the same kind of curatorial mindset the fourth edition of The Small Museums Cataloguing Manual advises:

“Cataloguing underpins many museum activities, including research, exhibition development, conservation, risk management, publication and outreach work, all of which are dependent on details and up-to-date collection of information.”

Truly, there is nothing new under the sun. That said, however, what’s in a catalog that isn’t in a label?

The location and history of the object, perhaps, possibly including labels used previously. The catalog for a vase in a museum, for example, might contain an entry along the lines of: Previously described as an Etruscan vase in the 1918 exhibition at The Royal Academy.

The policy on how the object is handled should also be included. For our vase, that might be: Must not be exposed to direct sunlight. Further clarification could also be required, like: Direct sunlight is defined as an excess of 10,000 lumens per square foot.

A description of the lineage or provenance of the object would add more value to the catalog, as would any other information that would help understand it, like a map showing where the Etruscan vase was made in ancient Italy.
If you’re wondering how this talk of a museum vase relates to the way organizations handle their data stores, this second quote from the cataloguing manual will explain:

"Documenting an object enriches its intrinsic value. The accumulation of information gives an object meaning and context, and results in a stronger understanding of its uniqueness, its contribution to the collection and its reason for being collected in the first place."

So the more detail we append to the data we store, the better we are able to protect it. Just like museum curators, however, we data professionals can’t spend all of our budget on protecting items that are of only trivial importance and easily replaced.

A Victorian clay pipe (my garden in London is full of them) might be worth exhibiting, but it’s not worth surrounding by high-tech defenses like those used for the Crown Jewels. Likewise, many organizations are reluctant to buy enterprise software for advanced encryption handling to enable developers to test database changes.

A sensible approach to making these decisions upfront is simply good use of time and budget. I might, for example, use coarse-grained labelling rather than fine-grained when I want to perform bulk operations in the name of efficiency.

Take a museum collection being packed in a shipping container to exhibit in another country. It needs to be labelled to determine the delicate transportation and handling it requires, based on a knowledge of the content, but that labelling is at the higher level of the shipping container, not the individual items in the collection. Similarly for a database, my backup and retention policy should be guided by specific data requirements, but applied at the database level (see the principle of minimization in the GDPR).

So what does all this mean for applying appropriate protection policies for databases? What goes in the catalog, and what in the label? How do they interact with each other? What are the challenges?

A common problem we’re hearing about with databases, for example, is that labels can’t be applied to the extended properties of a column, because the schema is not under the control of the DBA. The data is still her responsibility, but adding an extended property is changing the schema, which is either specifically prohibited as part of a support agreement, or vulnerable to conflicting schema changes when the vendor applies the next update.

Considering the analogous requirements for which the techniques of cataloguing were developed for museums, they really can help us to resolve issues like this and protect our data while controlling costs and implementation effort.

All of which makes the case for having both a catalog and labels for data compelling.

Policy should live in or near the catalog, stated with enough detail to resolve ambiguities like what ‘full daily backup’ actually means to our own organization. We should also be able to evaluate the impact of policy changes on the whole, like how many databases are running on availability groups, and how many have legacy versions nearing the end of support. History, lineage, and change over time properly belong at this level as well.

Labels have a special role to play as well. They travel with the object and provide a quick reference to inform the consumer (or user, or handler).

And finally, a word on lost or orphaned records from University College London’s museums and collections blog:

"In the event of theft, vandalism, fire or any other natural disaster, the worth of having the collection comprehensively documented (including photographed) is immense. In the case of theft, a precise list of objects can be presented to insurers and the police. Detailed collection information helps insurers value the museum’s loss and it aids police in investigating theft and, with luck, in identifying and returning stolen items. If an object were damaged, a thorough description may allow its restoration conservation. If it were lost, the catalogue would at least provide a record of its existence."

Apply the same thinking to data and suddenly the value of catalogs and labels becomes even more apparent.
Don’t just think database DevOps. Think compliant database DevOps.

The 2018 Accelerate State of DevOps Report from DORA specifically calls out database development as a key technical practice which can drive high performance in DevOps. It’s an interesting shift in our understanding of what should be included in DevOps workflows because it moves the database from being a bottleneck in the process to being a participant.

At the same time, however, new data protection laws are coming into play and consumers are more aware than ever before of how their privacy can be compromised. So how can the promise of releasing changes to the database faster and easier be balanced with the need to keep data safe and remain compliant with legislation?

The answer lies in going one step further than database DevOps and thinking about compliant database DevOps. The same broad workflow but one where protecting data is baked into the development process from the beginning by combining the agility of DevOps, the desire to include the database in DevOps, and the necessity to secure data throughout development.

There are four key areas where this compliant approach can help organizations deliver value faster while keeping data safe.
Introduce standardized team-based development

The way databases are developed has changed over the last few years. Once the sole responsibility of Database Administrators and database developers in siloed teams, application developers are increasingly expected to write the code for the database as well as the application.

Writing database code, however, uses a language like T-SQL, the syntax of which is not as strict as application languages like C#. Consequently, a number of different styles have emerged. Some developers, for example, prefer plain black type rather than seeing type in different colors, others hate indents, and arguments about whether commas should be at the beginning or the end of a line go on.

All of which can end up with the code behind a database being muddled and difficult to understand, particularly when different developers have worked on the same code base over time. Where teams of developers are updating a database repeatedly, they can collaborate much more easily if all of the code in the database is standardized in the same style.

Similarly, like application code, T-SQL has its own share of ‘code smells’ – errors in code which can affect performance, as well as those which could be breaking changes. Teams should also therefore adopt tools which use a static code analyzer to flag up potential errors and pitfalls in code as it is typed.

This can be particularly useful when first starting to code in T-SQL, or if there are specific rules that have to be followed by everyone on the team. It also provides a quality control gate at the point at which code is written so that before it is even committed, any potential issues have been minimized.

Probably the most useful and relevant team-based process to adopt is version control, which is standard in application development and involves developers checking their changes into a common repository frequently during the development process. As a direct result, everyone has access to the latest version of the code and it’s always clear what was changed, when it was changed, and who changed it.

This is just as true for the database code, which should also be version controlled, preferably by integrating with and plugging into the same version control system used for applications. By capturing the changes in version control, one source of truth can be maintained, reducing the chance of problems arising later in the development process, and also providing an audit trail of every change that has been made.

Automate deployments

Introducing version control to database development brings many advantages. Ad hoc changes and hot fixes are minimized, for example, reducing the chance the database will drift from its expected state. Every developer also works from the same code base, so there are fewer errors when different development branches are merged, and the audit trail of changes can be useful in demonstrating compliance.

A bigger reward awaits too because it opens the doors to the automation that DevOps encourages, while making the development process more secure.

Once version control is in place, continuous integration (CI) can be implemented to trigger an automated build as soon as code changes are checked in, which tests the changes and flags up any errors in the code. If a build fails, it can be fixed efficiently and re-tested, so a stable current build is always available.

A typical CI server in application development executes a series of commands that build the application. These commands clean directories, compile the source code, and execute unit tests. Where applications rely on a database back-end, the CI server can also perform the additional tasks of testing and updating the database.

This will generate an artifact which includes the deployment script required to update the database schema and any version-controlled static data, as well as additional information to make sure the release is safe. The artifact can then be tested by deploying it against a staging database and reviewing it to confirm it is production-ready.

Finally, it can be added to the same release automation tool used for the application, like Octopus Deploy or Azure DevOps, which provides another audit trail of the changes made and who approved them. This gives organizations the option to include the release of database changes with the workflow already in place for the application, rather than have to introduce a new, unfamiliar process.
Monitor performance and availability

It is normal practice to monitor databases to keep an eye on factors like memory utilization, I/O bottlenecks and growth trends. The increase in the size and complexity of SQL Server estates has already prompted many organizations to introduce third-party tools to give them a wider and deeper picture. Adding DevOps to the equation makes it even more important.

While the automation which DevOps introduces to many parts of database development minimizes errors and gives much better visibility across the whole process, there is a flipside. Instead of releasing changes to the database once or twice a quarter, changes can now be released at any time.

Even though they will probably be the small, iterative ones which DevOps encourages, there is still a chance they will cause problems when they are deployed, particularly if databases are under heavy load or there are differences between environments.

Given the importance the database has to many business operations, organizations should be able to spot queries having an unusual impact, deadlocks and blocking processes – and be able to drill down in seconds to the cause.

New data protection regulations also now require organizations to monitor and manage access to personal data, ensure data is identifiable, and report when any breaches occur. This makes an advanced monitoring solution a necessity in most cases, in order to monitor the availability of servers and databases containing personal data.

Given the added complexity it brings to monitoring, organizations should look for a solution that offers the extra capability but makes taking advantage of it easier. By, for example, allowing all SQL Server instances, availability groups, clusters, and virtual machines to be viewed on one central web-based interface. And by having customizable alerts that be configured to suit SQL Server estates of any size and complexity.

Protect and preserve data

Including the database in DevOps enables the full advantages of DevOps to be realized without the database being a counterweight. The new requirement for compliant DevOps, however, which requires data to be protected all the way through the development process, adds another factor.

Redgate’s 2018 State of Database DevOps Survey showed that 67% of developers want a copy of the production database in their development, test, or QA environments to ensure changes will work once deployed. This helps find problems sooner before they get to production, yet those same production databases invariably contain the sensitive data that needs to be protected.

This is where data masking tools which pseudonymize and anonymize data are now being adopted to provide database copies that are truly representative of the original and retain the referential integrity and distribution characteristics. Indeed, Gartner’s 2018 Market Guide for Data Masking predicts that the percentage of companies using data masking or practices like it will increase from 15% in 2017 to 40% in 2021.

While such masking can help when provisioning copies of production databases for use in development and testing, it can also lead to resource issues. It is not unusual for databases to be 1TB in size or more, and provisioning copies to multiple developers can take up a lot of time as well as space.

This is where the tried and tested virtualization technologies built into the Windows operating system come into place. SQL Clone from Redgate, for example, uses it to create copies, or clones, of databases in seconds which, while only around 40MB in size for a 1TB database, work just like normal databases and can be connected to and edited using any program. It can also be integrated with data masking to provision masked database copies with a process that is simple, fast, repeatable, transparent, and auditable.
DevOps has entered the conversation and provided a route to removing the database as the bottleneck in delivering value. By encouraging collaboration and integration, and moving to releasing small changes, often, database deployments change from worrying, infrequent problems to a standard part of the development and release workflows.

Data privacy and protection concerns have also interrupted the conversation with the GDPR now being joined by the upcoming Stop Hacks and Improve Electronic Data Security (SHIELD) Act in New York, the Consumer Privacy Act in California, and India’s Personal Data Protection Bill, among many others.

DevOps and data privacy do not need to oppose each other, however. Rather, they can complement one another. The automation and audit trails that DevOps processes introduce to database development can ease compliance with data protection regulations and enable organizations to balance the need to deliver software faster with the requirement to protect and preserve personal data.
The biggest driver for including the database in DevOps is to increase the speed of delivery of database changes. The biggest challenges are the perceived difficulties in overcoming different development approaches, so that the teams can synchronize application and database changes, and the disruption that is anticipated, as a result.

In this article, I’ll explain the different methods of getting the database under source control, either by storing per-object state, or migration scripts, or usually both. I’ll then describe how SQL Change Automation is the key that opens the door to including the database in continuous integration and automated release management.

In a typical automated deployment pipeline for applications, deployments are linked to source controlled versions of the application code so the changes made in development can be associated to what has been deployed.

Similarly, the database should also be source controlled, but there are some additional challenges. With a database, the data must persist — you can’t simply drop and replace the database, like you would an application.

When we update the database schema, for example altering a column’s data type, we must do so in a way that preserves the business data. This may require additional ‘migration scripts’, which must also be in source control, to enable existing data to fit in any changed schema. An example might be trying to deploy a change that added a NOT NULL constraint to a column with existing data. We’d need to write a migration script that loaded a default value to any NULL columns in the table, store that script in source control, and incorporate it in the automated deployment process.
**Loading static data and test data**

While your database is designed to store customer and transaction data, it will not function unless certain tables are “pre-stocked” with any immutable data required for dependent applications to function. It might be something as small and simple as a list of countries within which your organization can trade. This sort of lookup data, as well as seed data, enumerations, and more, all the data that makes your system work, needs to be stored in source control and deployed alongside schema changes.

You also need to think about how to provision realistic data for development and testing. How do you test the latest build in development with production or production-like data, for example, while ensuring that private and sensitive data is protected, in all environments?

**A working database system is more than schema-plus-static data**

A SQL Server database of any size will consist of many components on top of the SQL DDL code. As well as database-scoped objects and configuration settings, there are instance and server-level objects and settings to consider.

Special care needs to be given to the security configuration of any database. Certain objects (database users and roles) are database-scoped, but other users, roles and permissions are server-scoped. Ideally, you’ll separate out entirely your handling of users, roles and permissions, from the handling of code and schema changes.

To deploy a complete, and standardized database environment, you’ll need a configuration file, in source control, that provides a single reference file for all login and user details and the roles to which these users belong, and then another to define the permissions assigned to these roles.

In addition to security objects and static data, there are also likely to be agent jobs, external files, SSIS components, .NET libraries, PowerShell scripts, R and Python scripts and modules, and a host of other components.

Our ‘deployment package’, in source control, needs to contain everything that is essential for the working database, and to create the basic database and environment, reproducibly, on every target SQL Server instance, in both development and pre-production environments.

**Database drift**

How long does it take to make a one-line database change, run it through the normal testing process and get it to production? If your answer is measured in days, weeks or months, then you have a few problems. Firstly, delivery time for any significant new functionality will be slow. Secondly, when you hit a production issue, you won’t always have time to go back to the source code, make the fix, test it, and then redeploy. Instead, the DBA will often perform a hot fix, directly on the production database.

This drift causes inconsistencies that undermine tests and can cause failed deployments, either because code clashes, or because important fixes are accidentally rolled back.

**Database source control methodologies**

If you just want to tear down the current database version and build a new version then, leaving aside for a second all the “extra bits”, all you need in source control is the SQL DDL scripts describing the current ‘state’ of every database object (plus any required static data), and an automated means to create each database object in the correct dependency order.

However, what if you need to update an existing database? For database code objects, such as stored procedures and functions, your strategy is the same. For table changes, however, you may also need to store a set of change, or migration, scripts describing how to alter the metadata of the database, as defined by its constituent DDL creation scripts, from one database version to another, whilst preserving the data held within it.

So, what database artifacts should be in source control? The database DDL code, or the migration scripts? The answer is “both”, but different tools take different approaches.

A tool like SQL Change Automation (SCA) works directly in Visual Studio, and it auto-generates numerically ordered migration scripts, using the industry-standard SQL Compare engine, under the covers. Each of the changes, often ALTER commands, are then organized in SQL scripts and run in order at employment time, to migrate a database from one version to the next, while preserving any existing business data.

Alternatively, a tool like SQL Source Control, which is a plug-in for SQL Server Management Studio (SSMS), stores in source control the current state of each object in the database, as a CREATE script. SQL Source Control compares the set of CREATE scripts to the target database and auto-generates a script to synchronize the two states, again using the SQL Compare engine.

The two approaches are not mutually exclusive. Sometimes, you’ll need the per-object history and rollback advantages of the state-based approach. Other times, you’ll want the closer control over the deployment code or the dependency ordering that the migrations-based approach offers. If you use the state-based approach, you’ll occasionally need to handcraft correct change scripts to define the correct route for migrations that affect existing data. Tools that use the migrations approach often “back fill” the state of each object, at each version, so that we can see easily how the object has changed over time.

Also, it makes little sense to retain a string of change scripts of a code-based database. All you need to store in source control is a single script that creates the object if it doesn’t exist, or alters it otherwise, so that existing permissions on that object are retained.

**Automated deployments from source control with SQL Change Automation**

SQL Change Automation (SCA) is a tool designed to help you automate the way you build, test, and deploy SQL Server databases, from source control. It has the following components:

1. An SCA Visual Studio plug-in (previously called ReadyRoll) – to auto-generate database migration scripts for a source-controlled database
2. PowerShell Build components – to validate the database project and incremental scripts, create and test the build, and produce a build package you can publish
4. Add-ins – to extend some of the more common delivery tools such as TeamCity, Azure DevOps Services, MS Build, Bamboo, Jenkins, Octopus Deploy and TFS.
As discussed in the previous section, you can use SCA directly in Visual Studio, and it will auto-generate the correct deployment script from all the required change scripts. It also supports a state-based approach for source control of stored procedures, triggers, views, and functions. Alternatively, SCA will generate the deployment scripts from the latest versions of the CREATE scripts for each object, as defined by a SQL Source Control project. In fact, it will work with whatever is in source control, if the directory structure is compatible with SQL Compare.

Regardless of your starting point, once your database and application are in source control, the deployment pipeline is the same, and can use the same tools and techniques. SCA is designed, specifically, to adapt to your chosen development techniques, and to integrate smoothly with your existing toolchain, including source control, build, integration, and deployment tools. SCA is a PowerShell-based tool, making it easy to extend and adapt the processes that run as part of a database build or migration. Simply include the PowerShell scripts in the deployment package (a NuGet package), and run them pre- or post-deployment.

As can be seen, using SCA, the automated deployment pipeline for applications can, with a subtle tweak, also incorporate the database.

It sounds tempting, doesn’t it? The changes to both the application and database are committed to source control, which can then trigger a continuous integration process to test the changes, before they go through to release management, where the DBA can check the database deployment scripts before they leave development.

SCA will help address many of the database deployment challenges discussed earlier in the article. For example, SCA will:

- Wrap into a single NuGet package all the scripts you need to build or update a database schema, and insert any static data. A tool such as Octopus Deploy or PowerShell Remoting can then deploy the package across a network.
- Build or update the database automatically from the build script, migration script or object source.
- Validate the build – it will run pre-build syntax checks and can perform static code analysis. It will also integrate with a tool such as SQL Doc to document the database.
- Perform database testing – it can run tSQLt unit tests directly, and other tests using PowerShell scripting, and through integration with your CI Server.
- Automate deployments – you can run simple deployments directly within SCA, using PowerShell, and integrate with a release management tool such as Octopus deploy.

Finally, SCA is useful not just for moving database changes towards production, but also for provisioning individual developers with their own copies of databases to use in sandboxes, and for rapidly setting up test servers. SCA provides the validate and tested build, plus any ‘static’ data. We can then load test data, either from the NuGet package, or by integrating with a tool like SQL Data Generator, and deploy to each required server. If you need large volumes of test data, and to provision many test and development databases, then you can consider integrating with SQL Clone to speed up provisioning.
The real advantages of automating your release pipeline

Release Day used to mean weekends spent deploying the latest code, database changes, and website content. Release teams would huddle together in war rooms troubleshooting issues, deploying hotfixes, coordinating deployments, testing, fixing, and redeploying again and again.

As the trend in development shifts toward more agile delivery, and customers clamor for continuous deployment, this model of monolithic code releases begins to strain. Automating your release pipeline can cut down on your release cycle time and alleviate many of the Release Day issues that arise from still trying to manually deploy code.

So what benefits can you expect from automating your release pipeline?

Remove repetitive steps that are tedious and error-prone

Manually deploying any sort of content, whether it’s code, infrastructure changes, database scripts, or simple website content changes, can become tedious to perform repeatedly. This is a recipe for boredom and can lead to mistakes during deployment. These mistakes may even go undetected until users have started using the newly deployed changes. Automating these steps gives you a consistent, repeatable process that helps minimize your chance of a broken deployment.

Reclaim time spent manually deploying

Automating your release process doesn’t just alleviate boredom, it gives team members more time to spend on other activities that add value to your company. Time someone is spending pushing code out to websites and running tests is time they could be spending designing new features, fixing bugs, or having weekends with their families. Quite simply, automating your releases can reduce costs and make your employees happier.

Improve quality, make customers happy

As mentioned previously, automating your release pipeline can reduce the number of issues caused by human mistakes. This results in higher quality deployments with less chance for error.

Even with deployment steps completely automated though, time must still be spent making sure the most perfectly deployed code actually works, so automated testing should be included before and after deployment.

Automating tests help you detect critical path failures before your code lands in Production and makes your customers angry. It also further reduces the amount of time other team members are spending manually verifying that deployments were successful, rather than adding additional value to the company.

Turn on a dime

As your release pipeline becomes more streamlined, it becomes easier to adapt to changing business needs because you can quickly see what’s deployed where, and promote or abandon new builds with just a few clicks.

If a new feature isn’t well received after customers begin to use it, for example, improvements can be deployed and tested in hours rather than days or even weeks. Similarly, if a critical bug is found in newly released code, a well-architected release pipeline can deploy previous builds that are in a known good state and get your customers back online while your team works to resolve the issue.

This is a guest article from Nebbia Technology, a software company based in Orlando, Florida, which specializes in Azure-based solutions and DevOps best practices.

www.nebbiatech.com
One of the common things we find in enterprise organizations looking to move to a DevOps model is high levels of technical debt.

To be more accurate, they are caught in a vicious cycle of technical debt to the point that trying to ship anything in a rapid, agile way is nearly impossible. It’s the Greek debt crisis level of technical debt.

In many cases, layers and layers of process and management have been added into the software development lifecycle in order to try and fix the symptoms of the problem (low quality releases, bugs in production, unstable environments, poor performance, etc) but they are just Band-Aids on the underlying issues.

So how do we get out of this death spiral before the organization can’t compete any more and a disruptive innovator comes along and eats their lunch?
If we can automate some of the routine, error-prone and time-intensive tasks, we can leverage that productivity gain and invest the time freed up into repaying technical debt. As we pay back technical debt we get a higher quality, more stable and more agile application, and we can then reinvest in more automation to start the next cycle of improvement.

We know that this works because we’ve seen it and done it with clients. But it does come with a caveat or two.

Firstly, you need to get commitment from individuals like the Product Owner that the productivity gains will be spent on paying down technical debt and not an endless cycle of feature bloat (that was probably one of the causes of the problem in the first place).

I’m not going to wave a magic wand and say this is easy because it’s not. If you can find the right analogy, however (Technical debt is like walking through quicksand, or Technical debt is like trying to run a marathon with an 80lb backpack on, or Technical debt is what start-ups don’t have…) then you might have a chance.

Secondly, DevOps is more than just A for Automation – it’s Culture-Automation-Lean-Measurement-Sharing (CALMS), so ideally you’d do more than just automate some stuff and begin to start looking at being product-centric, coaching your product owner to understand operational requirements and moving away from the finance-driven project-centric model.

This is a guest article from DevOpsGroup, a UK DevOps consultancy which transforms and accelerates the way organizations deliver software

www.devopsgroup.com
CALMS is the acronym for a framework which allows businesses to assess how prepared they are on their journey to DevOps. CAMS (without the L) was first introduced by Damon Edwards and John Willis after the very first US-based DevOps Days in 2010. Jez Humble later added the L, standing for Lean, and the full meaning of the acronym is now:

**Culture**
Focus on people
Embrace change and experimentation

**Automation**
Continuous delivery
Infrastructure as code

**Lean**
Focus on producing value for the end user
Small batch sizes

**Measurement**
Measure everything
Show the improvement

**Sharing**
Open information sharing
Collaboration and communication

When talking about DevOps, culture is at the center of the transformation it can inspire. A DevOps culture changes the way businesses work, enabling teams to produce great results that make users happy.

A big part of the culture is breaking down the silos that exist within an organization, so that Dev and Ops can work better together towards the same goal – those happy users.

There’s no longer an *It worked on my machine or It’s not my problem* atmosphere.

Now, it’s *Let’s work together to make this the best experience for our users*.

It empowers teams to do what’s best for users, and ensure they can release as needed.

This is really important when it comes to the database. Traditionally, DBAs have been very siloed in their own departments, and have often been seen as a bottleneck to releasing changes faster. They took complete responsibility for the performance of the production database and the security of its data.

Now everyone – developers and DBAs – should be thinking about how to make the release as smooth as possible, how it will run in production, and where to build security in.
Automation

Automation is key when it comes to DevOps. If you want to release more frequently, the release pipeline is a perfect candidate for automation. It's completely repetitive and automation will help eliminate manual errors.

Another benefit of automation is the traceability it provides. You’ll be able to see exactly what changes were applied to which environments and when. You can even see who made the changes and why; maybe they relate to a user story or perhaps they were needed for a bug fix. If you have any manual approval steps in your pipeline, you’ll also be able to see who reviewed the changes and signed off the release.

Continuous Integration with a thorough set of automated tests is important for automating releases. The output of CI is a package with all the files needed for the release, which will be used to deploy to other environments, so you build once and deployments are consistent. Automated tests (unit, smoke, integration, performance, GUI) are also important to provide confidence in releases and catch any issues before they reach production.

Automation does NOT mean continuous deployment. In continuous deployment, every time a change is committed, it flows continuously through the release pipeline all the way to being deployed. In order to do this, you need to have a lot of trust in your automated tests.

Instead, I’m talking about continuous delivery, where automation is used to progress changes to the point where they can be released as needed, after being reviewed. This is increasingly common in application development and the process looks something like the workflow above.

As can be seen in the workflow, database development is a siloed process, but a lot of DBAs get scared when they hear about automating the releases of database changes to production.

But again, I’m talking about automation and continuous delivery, not continuous deployment. It’s also incredibly easy to include a review step as part of the automated pipeline so that DBAs can see exactly what’s going to happen before anything changes in production.

Include the database in continuous delivery and, rather than complicating the process, it actually looks a lot simpler,

It can also be far easier to introduce if the tools used for database development integrate with and plug into the tools already used for application development:

Here, database development is a natural partner in the deployment pipeline and continuous delivery for both the application and the database is achieved.

Another great thing about automation and databases is the ability to do a test deployment on a staging/pre-production environment that is as close to production as possible, automatically. This provides the best chance for success by testing the deployment script one last time before running it on production.
Lean

Lean was added after the initial talk and is an important value of DevOps. Lean is focused on incremental improvements and splitting the work into small batches. The small batches allow you to release frequently as you develop. This is important so that you can get real user feedback on your work and learn from it so that you can pivot and adjust according to what you learn as needed.

It’s hard to apply Lean to just the database. You should really care about the whole system. The database is part of that system. This further emphasizes the importance of including the database as part of the culture (breaking down silos) and the automation so you can have these frequent incremental releases to improve on.

Measurement

Measurement is important to get quick feedback and continuously improve, because you can only know if you’re making a difference and improving if you have a baseline to measure against. The first area to start measuring is your internal processes, like how long it takes for committed code to be running on production, the frequency of releases, the failure rate, and the mean time to recover from a failure.

You can also include telemetry – how your users are using your system, whether they discovered a new feature, and how it can be improved.

The second area to measure is the health of your system, and to understand that, monitoring is required. That way, you can react quickly if there’s a problem, or even better, proactively fix something that might become a problem, so you can keep your users happy.

For databases, measuring your internal processes and telemetry is done as part of the system since your database should be included in your release process. Monitoring databases allow you to ensure they are performing correctly, and it can be configured to provide alerts about performance issues or low disk space before they become problems.

Sharing

Sharing brings us back to the first value, Culture. If you want to make a huge impact on the culture, it’s important to share the lessons you’ve learned and your best practices with the rest of your company. This helps to break down those silos so that people work together to continuously improve and do what’s best for your users.

It’s also important for DBAs to share the findings with other DBAs and teams across the organization so they too can take advantage of all the benefits of including the database in their DevOps processes. That way, barriers will come down and DBAs will see the value of working on complex changes with development teams early in the development process, instead of finding problems when the changes reach production.

Conclusion

CALMS are important DevOps values, which enable you to break down silos, work better together, and provide frequent value to users that can be continuously improved. These values also relate to databases and it’s important to break down this silo as well so you can really fly.
Thanks to Redgate, you have the best tools.

We help teams deploy with confidence by automating infrastructure, code, and database delivery.

Contact us to go from “How do I use this?” to “Let’s do this!” faster.

www.nebbiatech.com | hello@nebbiatech.com | 407.930.2400
# Find a Redgate DevOps partner

## In Europe, Middle East, and Africa

<table>
<thead>
<tr>
<th>Company</th>
<th>Location</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture &amp; Performance</td>
<td>Lyon, France</td>
<td><a href="mailto:contact@architecture-performance.fr">contact@architecture-performance.fr</a></td>
</tr>
<tr>
<td>Axians IT Solutions GmbH</td>
<td>Ulm, Germany</td>
<td><a href="mailto:info@axians.de">info@axians.de</a></td>
</tr>
<tr>
<td>Coeo</td>
<td>Wokingham, England</td>
<td><a href="mailto:info@ceo.com">info@ceo.com</a></td>
</tr>
<tr>
<td>DATamasterminds</td>
<td>Amsterdam, the Netherlands</td>
<td><a href="mailto:info@datamasterminds.io">info@datamasterminds.io</a></td>
</tr>
<tr>
<td>DataMovements</td>
<td>London, England</td>
<td><a href="mailto:info@datamovements.co.uk">info@datamovements.co.uk</a></td>
</tr>
<tr>
<td>DevOpsGroup</td>
<td>Cardiff, Wales</td>
<td><a href="mailto:team@devopsgroup.com">team@devopsgroup.com</a></td>
</tr>
<tr>
<td>DLM Consultants</td>
<td>Cambridge, England</td>
<td><a href="mailto:enquiries@dlmconsultants.com">enquiries@dlmconsultants.com</a></td>
</tr>
<tr>
<td>GDS Business Intelligence GmbH</td>
<td>Essen, Germany</td>
<td><a href="mailto:info@geislers.net">info@geislers.net</a></td>
</tr>
<tr>
<td>Jarrin Consultancy</td>
<td>Newcastle-upon-Tyne, England</td>
<td><a href="mailto:contact@jarrinconsultancy.com">contact@jarrinconsultancy.com</a></td>
</tr>
<tr>
<td>Miracle</td>
<td>Reykjavik, Iceland</td>
<td><a href="mailto:info@miracle.is">info@miracle.is</a></td>
</tr>
<tr>
<td>N4Stack</td>
<td>Nottingham, England</td>
<td><a href="mailto:hello@n4stack.co.uk">hello@n4stack.co.uk</a></td>
</tr>
<tr>
<td>Nexus Technology</td>
<td>London and Jersey, UK</td>
<td><a href="mailto:enquiries@nexustech.je">enquiries@nexustech.je</a></td>
</tr>
<tr>
<td>Riley Waterhouse</td>
<td>Lancashire, England</td>
<td><a href="mailto:info@rileywaterhouse.co.uk">info@rileywaterhouse.co.uk</a></td>
</tr>
<tr>
<td>Sun City Software Kft.</td>
<td>Budapest, Hungary</td>
<td><a href="mailto:info@suncitysoftware.hu">info@suncitysoftware.hu</a></td>
</tr>
<tr>
<td>xTEN</td>
<td>Leeds, England</td>
<td><a href="mailto:info@xten.uk">info@xten.uk</a></td>
</tr>
</tbody>
</table>

## In North America

<table>
<thead>
<tr>
<th>Company</th>
<th>Location</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaptivEdge</td>
<td>Alameda, California</td>
<td><a href="mailto:sales@adaptivedge.com">sales@adaptivedge.com</a></td>
</tr>
<tr>
<td>Centino Systems</td>
<td>Oxford, Mississippi</td>
<td><a href="mailto:info@centinosystems.com">info@centinosystems.com</a></td>
</tr>
<tr>
<td>Computer Enterprises, Inc.</td>
<td>Pittsburgh, Pennsylvania</td>
<td><a href="mailto:info@ceiamerica.com">info@ceiamerica.com</a></td>
</tr>
<tr>
<td>Crafting Bytes</td>
<td>San Diego, California</td>
<td><a href="mailto:contact@crafthingbytes.com">contact@crafthingbytes.com</a></td>
</tr>
<tr>
<td>CSW Solutions</td>
<td>Chicago, Illinois</td>
<td><a href="mailto:info@cswsolutions.com">info@cswsolutions.com</a></td>
</tr>
<tr>
<td>InCycle Software</td>
<td>New York</td>
<td><a href="mailto:info@incyclesoftware.com">info@incyclesoftware.com</a></td>
</tr>
<tr>
<td>IowaComputerGurus</td>
<td>West Des Moines, Iowa</td>
<td><a href="mailto:sales@iowacomputergurus.com">sales@iowacomputergurus.com</a></td>
</tr>
<tr>
<td>Loring Consulting</td>
<td>Macungie, Pennsylvania</td>
<td><a href="mailto:brian@loringit.com">brian@loringit.com</a></td>
</tr>
<tr>
<td>Kingfisher Technologies</td>
<td>Cedar Rapids, Iowa</td>
<td><a href="mailto:info@kingfisherdata.com">info@kingfisherdata.com</a></td>
</tr>
<tr>
<td>Mi4</td>
<td>Houston, Texas</td>
<td><a href="mailto:blake@mi4.com">blake@mi4.com</a></td>
</tr>
<tr>
<td>LIKE 10 INC.</td>
<td>Ottawa, Canada</td>
<td><a href="mailto:info@like10.com">info@like10.com</a></td>
</tr>
<tr>
<td>Nebbia Technology</td>
<td>Orlando, Florida</td>
<td><a href="mailto:hello@nebbiotech.com">hello@nebbiotech.com</a></td>
</tr>
<tr>
<td>Northwest Cadence</td>
<td>Bellevue, Washington</td>
<td><a href="mailto:clientservices@nwcadence.com">clientservices@nwcadence.com</a></td>
</tr>
<tr>
<td>Plexifact</td>
<td>New York</td>
<td><a href="mailto:info@plexifact.io">info@plexifact.io</a></td>
</tr>
<tr>
<td>Polaris Solutions</td>
<td>Chicago, Illinois</td>
<td><a href="mailto:info@polarissolutions.com">info@polarissolutions.com</a></td>
</tr>
</tbody>
</table>

## In Latin America

<table>
<thead>
<tr>
<th>Company</th>
<th>Location</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Watchmen</td>
<td>Austin, Texas</td>
<td><a href="mailto:info@sqlwatchmen.com">info@sqlwatchmen.com</a></td>
</tr>
<tr>
<td>SQLCore</td>
<td>São Paulo, SP</td>
<td><a href="mailto:comercial@sqlcore.com.br">comercial@sqlcore.com.br</a></td>
</tr>
</tbody>
</table>

## In Asia Pacific

<table>
<thead>
<tr>
<th>Company</th>
<th>Location</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBInsight</td>
<td>Brisbane, Australia</td>
<td><a href="mailto:rob@dbinsight.com.au">rob@dbinsight.com.au</a></td>
</tr>
<tr>
<td>Human Interactive Technology Inc.</td>
<td>Tokyo, Japan</td>
<td><a href="mailto:rg_support@hit-inc.jp">rg_support@hit-inc.jp</a></td>
</tr>
<tr>
<td>LobsterPot Solutions</td>
<td>Melbourne, Australia</td>
<td><a href="mailto:contact@lobsterpot.com.au">contact@lobsterpot.com.au</a></td>
</tr>
<tr>
<td>SQL Down Under</td>
<td>Melbourne, Australia</td>
<td><a href="mailto:info@sqldownunder.com">info@sqldownunder.com</a></td>
</tr>
<tr>
<td>SQL Maestros</td>
<td>Bangalore, India</td>
<td><a href="mailto:info@redgate.sqlmaestros.com">info@redgate.sqlmaestros.com</a></td>
</tr>
<tr>
<td>SQL Masters Consulting</td>
<td>Brisbane, Australia</td>
<td><a href="mailto:warwick@sqlmastersconsulting.com.au">warwick@sqlmastersconsulting.com.au</a></td>
</tr>
</tbody>
</table>

We have a growing number of consultants across the globe who can help you with the installation, customization and training of our products to get you on the road to compliant database DevOps.

If you’re interested in becoming a Redgate Certified Partner, or would like to find out more, visit:

[www.red-gate.com/consultant](http://www.red-gate.com/consultant)
Compliant Database DevOps
Deliver value quicker while keeping your data safe

Standardize
team based development

Automate
database deployments

Protect
& preserve data

Monitor
performance & availability

Redgate helps IT teams balance the need to deliver software faster with the need to protect and preserve business critical data.

You and your business benefit from a DevOps approach to database development, while staying compliant and minimizing risk.

Find out more at
www.red-gate.com/DevOps