WHITEPAPER

Compliant Database DevOps and the role of DevSecOps

redgate

Executive Summary

DevOps is now widely accepted in software development because, by introducing a culture of collaboration and cooperation between development and IT operations teams, it enables features to be released faster to end users.

It works too. The 2017 State of DevOps Survey from DORA and Puppet shows that companies which adopt DevOps deploy changes 46 times more frequently, have a change failure rate that is five times lower, and are able to recover from breaking changes 96 times faster.

Now, however, DevSecOps is entering the picture because it balances the need for companies to deploy changes to code more frequently with increasing demands for the same code to be secure.

In its October 2017 report on 10 things to get right for successful DevSecOps, Gartner predicts that by 2021, DevSecOps practices will be embedded in 80% of rapid development teams, up from 15% in 2017.

But how can the database join the conversation? Can the same principles be followed, or should they be modified? How can increasing regulatory pressure around data privacy and protection be satisfied? What additional measures should be considered so that the security of data can be protected alongside the code and truly compliant Database DevOps is achieved?

This whitepaper details how and why DevOps can be applied to the database, and then outlines the steps necessary to include it in DevSecOps.

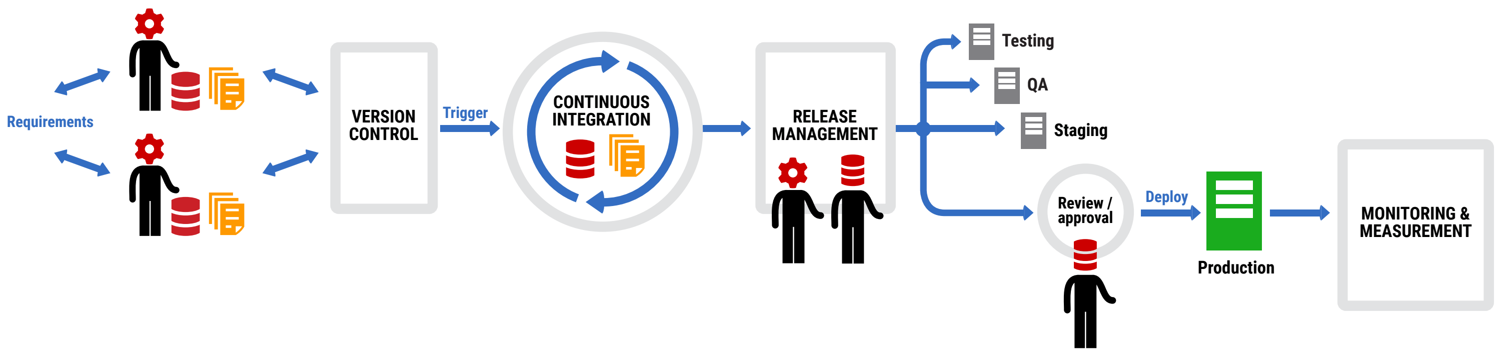
Contents

Including the database in DevOps

Once the preserve of big companies like Amazon and Facebook, Google and Netflix, DevOps has moved into the mainstream and is now being widely adopted. The 2018 Database DevOps Survey from Redgate revealed that 52.5% of companies have already adopted a DevOps approach to some or all of their projects, and a further 30% plan to do so in the next two years.

DevOps involves making small changes and deploying them often, rather than relying on big bang releases. It also introduces testing much earlier in the development pipeline to the point when changes are made, rather than deployed. That way, any breaking changes can be identified immediately, rather than at release time when fixing them will be more difficult and take a lot more time. And finally, DevOps automates as much of the development process as possible to make it reliable, repeatable and consistent.

The database was not originally part of the DevOps conversation because it was the preserve of Database Administrators (DBA) and database developers, a different coding language is used, and it protects valuable customer and transactional data that has to be protected and preserved at all costs.



DevOps itself, however, requires the database to be included because the faster development of applications means databases also need to be updated more frequently. Evidence of this emerged in the 2018 Database DevOps Survey mentioned earlier, which showed that 76% of application developers are also now responsible for database development. They have been brought into the database fold because front-end and back-end development are now much more closely connected.

With separate tooling and different development processes, however, the deployment of database changes often becomes a bottleneck, hindering the speed of change which DevOps can otherwise bring.

Hence new software has emerged which integrates with and plugs into the same infrastructure used for developing applications. As shown in the diagram above, this streamlines database development and enables it to be included in processes like version control, continuous integration and release management, making deployments faster, easier and far less prone to errors.

Importantly, the standardization and automation it brings to many aspects of development also lends itself to a Compliant Database DevOps approach where regulations and concerns around data privacy and protection can be addressed with relative ease. The masking of sensitive data, for example, can become a routine part of the process, and the audit trail of changes made and deployed can demonstrate compliance at every stage of development.

Adding DevSecOps to the equation

DevSecOps emerged in 2012 as a direct consequence of software development teams beginning to take up DevOps. It had to because DevOps changed the rules of the game.

Before DevOps, application release cycles were typically every three or six months, sometimes yearly. Time was scheduled at the end of each cycle for the security team to review and certify the release, recommend changes where necessary, and it was an effective, if slow, process.

DevOps speeds up the pace of releases dramatically and even back in 2011, Amazon was on record as deploying changes every 11.6 seconds. Most companies don't need to match that, but with weekly, daily and hourly releases becoming the norm, the way security teams work had to change.

This was picked up by Gartner analyst, Neil MacDonald, in a 2012 blog post **DevOps Needs to Become DevOpsSec**. He argued that, contrary to popular opinion at the time, DevOps was not at odds with security concerns, and could become part of it with what he called DevOpsSec:

> *"DevOps must evolve to a new vision of DevOpsSec that balances the need for speed and agility of enterprise IT capabilities with the enterprise need to protect critical assets, applications and services."*

What was to become DevSecOps was born and the very way security teams engage with development teams changed. Instead of being the gate at the end of the development pipeline, security started to become baked into the pipeline with tools like static code analyzers and open source code vulnerability scanners testing code as soon as changes are made.

Neil MacDonald has continued to research the DevSecOps space and in 2017 co-wrote the Gartner white paper, 10 things to get right for successful DevSecOps. This was prompted by an increasing number of Garner clients asking about the topic, and the white paper suggests enterprises take the following steps to protect applications during development:

1. Adapt security testing tools and processes to the developers, not the other way around
2. Quit trying to eliminate all vulnerabilities during development
3. Focus first on identifying and removing the known critical vulnerabilities
4. Don't expect to use traditional testing techniques without changes
5. Train all developers on the basics of secure coding, but don't expect them to become security experts
6. Adopt a security champion model and implement a simple security requirements gathering tool
7. Eliminate the use of known vulnerable third party components
8. Secure and apply operational discipline to automation scripts
9. Implement strong version control on all code and components
10. Adopt an immutable infrastructure mindset

Think of these steps in terms of database development and they are all relevant apart from the second. Given that many databases contain personal data, some of which is sensitive, every vulnerability during development should be addressed.

Adapting DevSecOps for the database

Just as the rising number – and scale – of network breaches and hacks is prompting more and more companies to integrate security into their application development, so it must also be included in database development. Database DevOps must become Compliant Database DevOps

It's been hiding in plain sight too. Application developers typically use sources like the Ten Most Critical Web Application Security Risks from the Open Web Application Security Project (OWASP) to keep them updated on threats they should be aware of. In the latest list from 2017, sensitive data exposure comes in at number three:

> *"Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII … Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser."*

The introduction of data protection regulations in Europe and California,, along with data breaches like the debacle with Facebook and Cambridge Analytica, have also raised the awareness – and importance – of data privacy and protection.

That said, Gartner's recommendations for succeeding with DevSecOps can be applied to database development by following four steps.

1. **Identify the vulnerabilities**
2. **Remove the vulnerabilities**
3. **Introduce secure coding**
4. **Automate where possible**

Identify the vulnerabilities

At the center of DevSecOps for the database is the data, which has to be protected from both internal threats like negligence or accidental exposure, and external threats like hacking.

The biggest problem with data, however, is its size and sprawl. Firstly, data is growing at an annual rate of 58% according to IDC's 2017 Data Age 2025 white paper. And secondly, it is stored in a multitude of different places like production and development servers, remote backups, and the databases of different departments, business units and satellite offices.

So in order to identify what the vulnerabilities are, the data needs to be located and classified, and its risk identified. Data mapping exercises like this can be done by hand or with a third party tool, but it's important to create a record of every database, including copies used in development or testing, to gain a real understanding of where data is stored and who has access to it.

Classifying the data can then be undertaken to identify which data is sensitive and at risk, like credit card details, and therefore needs to be protected.

Faster data mapping

Many organizations find third party tools can help them map their server estate and classify their data.

By replacing the manual effort of finding what data is stored where, and what kind of data it is, they can streamline the process and introduce a standard, organization-wide method that can be followed by everyone.

Redgate's SQL Estate Manager, for example, helps organizations to map out their SQL Server estates through the auto-discovery of servers and the ability to assign labels to tables and columns to describe the sensitivity of the data they hold.

Remove the vulnerabilities

We saw earlier that Gartner recommends not trying to eliminate every vulnerability in application development. This is valid because runtime protection controls like intrusion prevention systems and web application firewalls also exist, and addressing every small concern would hinder development.

This isn't the case with data. Where data has been identified as sensitive and at risk like credit card details, access must be limited and it cannot be widely shared.

Developers, however, often need a copy of the production database in their development, test, or QA environments in order to ensure changes will work once deployed. The 2017 Database DevOps Survey revealed this was the case in 67% of organizations, yet those same production databases invariably contain sensitive data that needs to be protected.

One solution is to have a version of the production database with a limited dataset of anonymous data which is always used to develop and test against. This does, though, mean changes are tested against a database that is neither realistic, nor of a size where the impact on performance can be assessed.

Simple, repeatable data masking

Having realistic data available for development and testing is an important element in ensuring breaking changes do not reach production.

A third party tool can help in masking database copies, yet also providing development teams with test data that looks – and behaves – in the same way as real data.

Redgate's Data Masker offers multiple masking methods, includes replacement data sets, and provides a simple, repeatable, push-button process which can be quickly applied whenever a test database is refreshed with production data.

Another solution is to take a copy of the production database and mask the data manually by replacing columns with similar but generic data. This copy can then be used in development and testing, but will age very quickly as ongoing changes are deployed to the production database.

This is where data masking measures like pseudonymization, encryption, anonymization and aggregation should be adopted, and using a third party tool would significantly ease this process.

Introduce secure coding

It was mentioned earlier that a different coding language is used to write the code to create, maintain and update databases. While application development uses a language like C#, database development relies on a language like T-SQL, which is used for SQL Server.

And just as application developers have different styles and preferences when writing code, so the way T-SQL is written varies across development teams. It can be formatted differently, structured in a different way, and while some developers will be experienced in using T-SQL, it will be new to others.

With the T-SQL being updated at varying times by several people, this can result in the code being confusing, hard to understand, and containing errors that can cause deployments to fail.

Effortless, shareable SQL coding

A third party tool like Redgate SQL Prompt can help when introducing consistent, secure coding standards across teams. It auto-completes T-SQL as it is written, and real-time analysis identifies errors instantly.

It also offers customizable and shareable code formatting, and a library of code snippets which can be used across teams to introduce consistency.

Ideally, code should be written in the same style, good code that is often repeated should be shared across teams, code from third party sources should be avoided, and static code analysis should be used to highlight errors or bad code as soon as it is written.

Key to any approach like this will be the involvement of the Database Administrator, who will be the most conversant with T-SQL coding, and who can set the standards which should be followed.

This secure coding approach is essential when introducing DevOps to the database to ensure the deployment of changes is reliable, repeatable, and consistent. One notable company, Skyscanner, moved from deploying database changes once every six weeks to deploying them 95 times a day, for example. Even with less frequent deployments, introducing a secure coding mindset at the point code is written will prevent problems later down the line.

Automate where possible

The ninth point in Gartner's list is to implement strong version control on all code and components.

Version control is becoming standard in application development and involves developers checking their changes into a common repository during the development process, preferably at the end of each working day. As a direct result, everyone has access to the latest version of the application, and it is always clear what was changed, when it was changed, and who changed it.

This is just as true for the database code, which can also be version controlled, preferably by integrating with and plugging into the same version control system used for applications.

Compliant database development

When introducing DevOps and DevSecOps to the database, third party tools can ease the process.

Redgate's SQL Toolbelt includes the industry-standard tools for SQL Server development and deployment, and can be used to introduce version control, continuous integration, and automated deployments.

Interestingly, the 2017 Data of DevOps Report mentioned earlier also found that organizations which adopt DevOps processes like version control spend 50% less time remediating security issues, so the tenets of DevOps already strongly align with DevSecOps.

Perhaps the most important point here is that once version control is in place, it makes the automation that DevOps encourages possible, and means the whole development process is more secure.

Every time a change is committed to version control, for example, a continuous integration process can be triggered which tests the change and flags up any errors in the code. The errors can be fixed immediately and tested again, before the change is then passed along to a release management tool where the change can be reviewed before it is deployed to production.

In this way, the same discipline can be applied to every process that is automated, and all code changes are tested before they are deployed to ensure the production environment is never compromised.

Summary

Adopting DevSecOps for the database is just as appropriate – and essential – as it is for applications. Gartner's top ten list can also be used as the blueprint for how it can be achieved, with the one exception that every vulnerability during development does need to be addressed because the security of data is paramount.

The different measures required, like mapping data estates to identify vulnerabilities, masking sensitive data in use during development to remove vulnerabilities, and introducing secure T-SQL coding to prevent new vulnerabilities, will ensure DevSecOps for the database can be achieved alongside the application:

	Application DevSecOps	Database DevSecOps
1	Adapt security testing tools and processes to the developers, not the other way around	
2	Quit trying to eliminate all vulnerabilities during development	Identify the vulnerabilities
3	Focus first on identifying and removing the known critical vulnerabilities	Remove the vulnerabilities
4	Don't expect to use traditional testing techniques without changes	Introduce secure coding
5	Train all developers on the basics of secure coding, but don't expect them to become security experts	Introduce secure coding
6	Adopt a security champion model and implement a simple security requirements gathering tool	Identify the vulnerabilities
7	Eliminate the use of known vulnerable third party components	Introduce secure coding
8	Secure and apply operational discipline to automation scripts	Automate where possible
9	Implement strong version control on all code and components	
10	Adopt an immutable infrastructure mindset	

Importantly, the same version control and automation processes that are used in application DevOps can also be introduced to database development, so that DevSecOps is maintained.

The only question remaining is how to adapt security testing tools and processes to developers. There are many third party tools available, and we would recommend choosing those which integrate with and plug into the infrastructure already in place.