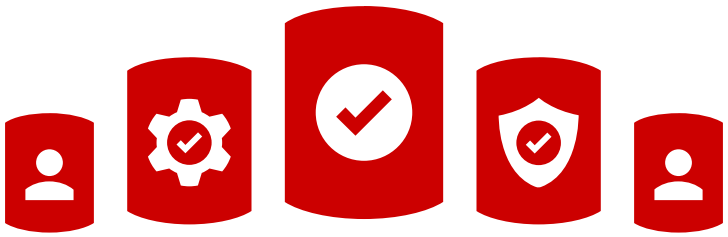
WHITEPAPER

# Essential practices for high performing Database DevOps teams

Compliant
Database DevOps

redgate



# Essential practices for high performing Database DevOps teams

# Introduction

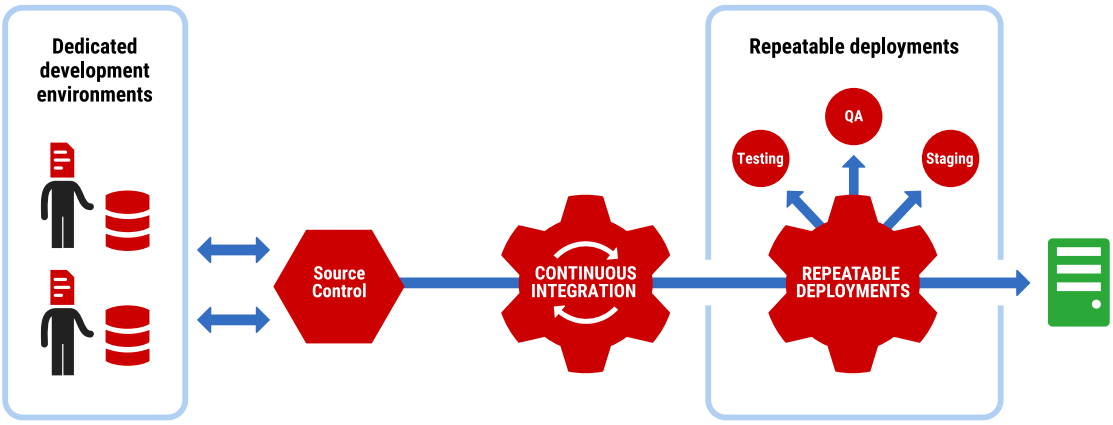
To enable high performance, teams should implement a core set of industry best practices to achieve Continuous Delivery also in database development. This 'kernel' of Database DevOps leads to better alignment between database and application teams and increases the throughput of higher quality releases.

As a result, software teams can meet customer demands quicker and gain a competitive advantage over organizations where the database is a bottleneck.

This whitepaper describes four key practices for database teams to adopt when evolving their database development process to follow a DevOps approach. For teams who are already following a DevOps approach it serves as a way to check that best practices are being following.

**The four essential practices for high-performing Database DevOps teams are:**

- Dedicated development environments
- Version control
- Continuous integration
- Repeatable deployments



# Dedicated development environments

One of the most obvious differences between application and database development is the split in how developers are organized to carry out their work. It's unheard of on an application team for developers to work on a central codebase without local versions, yet 46% of developers work in this way when it comes to database development.

Having a local, private database to work with empowers developers to try out new approaches without impacting the work of others. They provide developers with sandboxes in which to run experiments safely and test code in isolation before sharing it with the rest of the team.

**Dedicated development environments are foundational for agile development and DevOps. But moving toward this way of working is not a trivial process for database teams that need to balance their resources in terms of time, storage cost, and information security. The following techniques, however, help to facilitate the process:**

- **Data virtualization**
- **Data masking**
- **Automation and self-service**

## Data virtualization

In comparison to application code, databases are large and complex, and equipping every developer with their own personal copy of the production database could mean allocating significant storage. For many organizations this alone blocks dedicated database development environments. Data virtualization technology should be employed to reduce the size of database copies and make efficient use of infrastructure resource.

> "Traditionally, database work was the one area where there was less freedom to run quick experiments. Now, suddenly, it's very easy to create two clones, side by side, when I'm working on two different projects, or wanting to try out two different ways of doing something. I think going forward that sort of freedom is going to change the way the team approaches database work, and allow us to deliver changes much more quickly."

**Daryl Griffiths**, Principal Software Engineer, Moody's Analytics

Data virtualization results in lightweight database clones that can be provisioned to developers quickly without requiring large amounts of storage. Virtual clones work just like normal databases – they can be developed on and tested, and their updates deployed upstream – but their tiny storage footprint makes them incredibly agile and cost effective. Because the master dataset they reference can be a complete copy of the production environment, each clone created from it is fully representative and referentially intact.

Development teams who can work with production datasets see an increase in productivity and a reduction in data-related code defects. Realistic datasets enable developers to validate their work at the earliest possible opportunity, test whether they produce the desired results, and estimate any impact on performance. By shifting testing left in this way, any issues can be fixed before changes are shared with other developers or deployed to the next testing environment. . Reducing multi-TB databases to tens of MBs also enables developers to have many local copies for greater flexibility in testing.

## Data masking

With realistic test data, developers can take greater ownership of their changes, and not need to rely on others picking up bugs later.

However, spreading copies of production data around development and testing environments risks non-compliance with data privacy regulations, and leaves organizations vulnerable in the event of a data breach.

Sensitive data should be de-identified, or masked, before being delivered, so that developers can work with datasets that accurately reflect production without compromising information security and compliance. Masking data after it's been copied to development environments isn't recommended because wherever confidential information lives, even for a short while, there's the potential for unauthorized access or an accidental leak to occur.

**"Test data (or copy data) virtualization is a technology that is increasingly popular, when used in combination with 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."**

**Gartner 2018 Market Guide for Data Masking,**

Marc-Antoine Meunier, Ayal Tirosh, July 20, 2018

## Automation and self-service

Increasing the number of developer databases adds additional overhead for DBAs when it comes to refreshing environments with the latest test data. Data virtualization may enable databases to be refreshed in seconds, but responding to tickets for new data still adds to an already busy to-do list. Automation should be used to deliver fresh data to developers and keep development environments in sync with production.

Data virtualization also opens opportunities for greater agility within the team by way of self-service. The images from which virtual copies are created provide a single source of truth so that developers only receive data as intended – whether this is masked for compliance or specific to their project.

By establishing this control, it's safe to let developers pull a copy of the database they need on demand. This gives developers the freedom to run their own tests and then quickly reset the database back to the original state, improving the quality of the software in a shorter period.

## Key takeaways

**Data virtualization allows developers to have multiple local copies without placing a demand on infrastructure resources.**

**Data masking enables developers to work with realistic production datasets without risking noncompliance.**

**Automation and self-service remove bottlenecks to refresh test data and are enabled by data virtualization and data masking.**

## Redgate solutions that can help

SQL Provision

**SQL Provision** enables dedicated database environments and fast delivery of compliant data for development, test, and CI.

Data Masker

**Data Masker** allows you to replace sensitive data with realistic, anonymized test data.

# Version control

Dedicated environments empower developers to experiment and seek out better solutions to problems, leading to higher quality releases and faster throughput of changes as a result of early testing. Version control must be implemented to manage multiple changes happening in parallel across the development team.

Version control systems have long been an established part of the application development process but for database teams it's a more recent addition to the toolkit. Redgate's 2019 State of Database DevOps Report found that 45% of database environments lack version control, compared with only 17% for application code.

**Version control is foundational for Database DevOps, and helps to advance software development by:**

- **Aligning database and application code**
- **Improving visibility, compliance, and auditing**
- **Establishing a single source of truth**

## Aligning database and application code

Standardizing development practices and processes helps to improve quality and increase deployment frequency for software teams. An important step forward in standardization is to store database code and modification scripts in version control along with application code. This provides teams with the foundation for collaborating on software updates, enables changes to be merged more easily, and facilitates better coordination between teams when troubleshooting issues

## Improving visibility, compliance, and auditing

Version control provides a view into development work, its progress, who's doing it, and why. It also maintains detailed change histories and can be integrated with issue tracking and project management systems. Because the history it provides is incremental, version control lets developers explore different solutions and roll back safely in the case of errors while maintaining the referential integrity of the database.

Another important benefit of having an automatically generated history of changes is that it forms an audit trail for database code. This information is useful in the event of a data breach or compliance audit.

## Establishing a single source of truth

Version control manages the merging of changes from developers to establish a 'single source of truth' that can be deployed to testing environments and eventually released to production. The ability to merge changes easily is especially important for team members working in dedicated environments or based in different locations.

Branching features in version control enables developers to experiment in an isolated way, so that their changes don't impact others when they're under early development. A dedicated development environment is needed for database changes because using a branch with a shared database environment exposes the changes to others.

By determining a single, stable version of the database, complex processes become easier to automate and repeat, and deployments more predictable. Synchronizing application and database changes also becomes easier, enabling database and application code to be tested together so that issues are resolved earlier.

## Key takeaways

**Synchronizing application and database code speeds up the development and testing cycle leading to higher quality releases.**

**Version control provides an audit trail of database development, providing a detailed history of who changed what, when, and why.**

**Version control enables multiple changes to be merged easily when developers are working in dedicated environments.**

## Redgate solutions that can help


**Redgate Deploy** offers a versioned migration framework for 20 relational database systems as well as object level versioning and deployment tracking / auditing for SQL Server and Oracle.

# Continuous integration

Transitioning to dedicated development environments and implementing version control are the two processes that set the foundation for Database DevOps. They empower developers to elevate the quality of their work and provide control and visibility for DBAs.

With this foundation in place, automated testing and validation is now possible through the application of continuous integration (CI). The purpose of the CI process is to ensure that a build, test, and upgrade package process is performed in a known and repeatable fashion. Every change checked into the version control system results in an automated CI process execution that provides rapid feedback to developers, allowing fixes to occur immediately and successful builds to progress along the release pipeline.

**A CI process consists of the following three stages:**

1. **Building the database from scratch**
2. **Automated testing**
3. **Capturing upgrade scripts**

## Building the database from scratch

Often when code is executed on a development database, it's to build objects for use by an application. However, due to refactoring, the assembly of these objects might not be accurate, since other dependent objects may no longer work correctly. It's also common that a last-minute change to complex database scripts results in invalid code.

A build process for a database should involve rebuilding all objects, allowing dependencies to be resolved and checked by the database management system. These objects can be built from scratch in a new database, as we recommend at Redgate, or changes can be applied to a database at a known state while tests are executing against all other objects.

In either case, the process should detect any issues with building the database. If it does, the CI process will fail. If the process succeeds, developers can be confident that the database will be structurally intact and can be rebuilt from scratch.

It's recommended to commit code into version control frequently to speed up the CI process and break updates down into smaller chunks to reduce the number of conflicting changes.

## Automated testing

The next stage of a CI process involves running automated tests. These tests may be no different to the tests developers run on their own machines, but the benefit with using a CI process to run them is that the tests are always executed.

A CI process removes the risk of forgetfulness on the part of a busy developer, and ensures all the necessary tests are executed every time against all the code. This contrasts with most manual testing, which often limits testing to the scope of the objects being changed, ignoring the potential cascading effects on other pre-existing objects.

Data virtualization is incredibly beneficial to the CI process as it enables production copies to be spun up without the infrastructure overhead. By mirroring testing environments with production, data-related issues can be discovered earlier, leading to higher pass rates for releases.

## Capturing upgrade scripts

The final step of the CI process is the production of a package that can be executed against test, QA, staging, UAT, and ultimately, production environments. This package should be captured and stored without changes and used as the basis for testing updates in different non-production environments. If further testing uncovers problems, these should be fixed in the development environment, committed to version control and the entire CI process rerun.

There are numerous ways to produce this package, but the assembly of the script should be through an automated process that ensures the package is produced in the same way each time the CI process runs.

## Key takeaways

A CI process automatically validates schema changes so developers have confidence the database will be structurally intact and can be rebuilt from scratch.

A CI process ensures all necessary tests are executed every time against all the code and removes the risk of human error.

A CI process results in the production of a package that can be executed against testing environments and ultimately released to production.

## Redgate solutions that can help


**Redgate Deploy** offers advanced features to enable Continuous Integration of database changes. It is also integrated with the most popular application CI tools such as Octupus Deploy, Teamcity and many more.


**SQL Provision** creates lightweight database clones to validate changes as part of this process.

# Repeatable database deployments

Deploying database changes to new environments is a is a critical activity with inherent risks. Many development teams dread deploying changes to a production database because of the potential for problems or downtime.

By deploying database updates to testing environments alongside application code, errors are uncovered long before they reach customers. And the more environments a deployment of a potential release candidate can be tested on, the greater the chance that those same changes can be deployed successfully to the production database.

Standardizing the configurations of database environments along the release pipeline and automating the steps to deploy software ensures that the process is performed consistently every time. Each successful deployment to a new environment increases the confidence and likelihood of a successful release to production.

It's essential that a release management tool is used to control this process and ensure that the packages produced by the CI process are tracked. This way, if a package deployed to a test environment passes, the same package can be deployed to further environments. If the package fails, a new one is created through the development, to version control, to CI process. Any issues with the deployment process should be logged and captured, so that they're available for development and operations teams to review and correct.

## Key takeaways

**Automating database deployments to testing, QA, and other environments enables changes to be fully checked before risking a release to production.**

**Standardizing the configurations of database environments throughout the release pipeline and automating the steps to deploy software ensures that the process is performed consistently.**

**Automated deployments should be facilitated by a release management tool which tracks packages produced by the CI process.**

## Redgate solutions that can help


**Redgate Deploy** integrates with release management systems to automate database deployments safely with built-in review steps.

# Customer success stories

PASS

PASS no longer faces issues with merge conflicts, and have transformed deployments from long, worrying evenings to a smooth, error-free process.

www.redgate.com/PassCaseStudy

Moody's Analytics

Developers at Moody's Analytics can self-serve database copies, which has contributed significantly to the speed and efficiency of their database development and testing processes.

www.redgate.com/MoodysAnalytics

# Summary

The essential practices that high-performing Database DevOps teams follow are: dedicated development environments, version control, continuous integration, and repeatable deployments. By adopting these four practices, database teams can align their development and release process with application teams to increase throughput and success rates.

Dedicated development environments are foundational for DevOps but making the switch isn't trivial for database teams. Data virtualization should be used to overcome challenges with disk space and refresh times. Combined with data masking to deliver compliant test data to developers, this opens opportunities for automation and self-service as well.

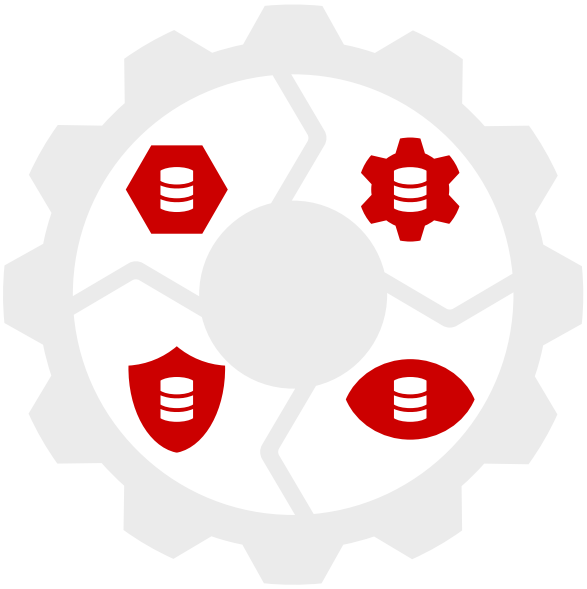
Version control has long been an established part of the development process for application teams, but for database teams it's a more recent addition to the toolkit. It's necessary for merging changes and provides an audit trail of database updates for visibility, troubleshooting, and compliance. Version control aligns database and application code and provides a single source of truth for continuous integration and database delivery.

With dedicated development and version control in place, database teams are able to make and share changes faster. It's only natural, then, that automation should be used to provide fast feedback on their work. Continuous integration takes the code in version control and runs it through an automatic build and testing process, allowing fixes to occur immediately and successful builds to progress along the release pipeline. The output of this entire process is a package that can be deployed to test, QA, staging, UAT, and ultimately, production environments.

Getting this all set up will take some work, but the pay-off is the ability to release updates faster with a higher degree of success – not just for the database, but for the software as a whole. To succeed in today's digital arena, organizations must be set up to deliver ongoing value to customers. Those who implement Database DevOps are better able to gain a competitive advantage, especially against organizations that experience a bottleneck in their release cycle caused by the database.

**"With techniques such as Continuous Delivery becoming more mainstream, automated database migrations are a baseline capability for many software teams."**

**Erik Dörnenburg, Head of Technology Europe**



# Redgate's Compliant Database DevOps solution

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

Your business benefits from a DevOps approach to database development, while staying compliant and minimizing risk.

Redgate's Compliant Database DevOps solution gives you an end-to-end framework for extending DevOps to your database while complying with regulations and protecting your data.

www.redgate.com/DevOps