# DevSecOps:
## Putting Database Sec in DevOps

DBmaestro
DevOps for Database

# Introduction: Why DevSecOps?

Modern practices, increasing agility, cloud development, and DevOps are shifting the focus from the traditional perception of security. DevSecOps is a set of practices that help the industry keep pace with innovation while making sure everyone is responsible for security, and that it is dealt with as early as possible.

Trying but failing to keep up by applying slower, traditional security measures to an agile process or a frequently changing cloud environment, often towards the end of the development process, is a waste of time. Instead, a more agile system is needed, one that is applied at all lifecycle stages—from conception to implementation, deployment, and maintenance.

Complementing both Dev and Ops, and often bundled with regulatory compliance, Sec should become a bridge to help move from a slower, traditional security approach to a modern, cloud enabled, high frequency, high capacity practice.

# The benefits of implementing DevSecOps are clear

## Secure application delivery

Delivery of a more secure application at a faster and more frequent pace via a continuous integration and testing process:

- Security teams can become a part of the early development process. They can comment, request, or change designs and strategies to make sure security issues don't need to be retrofitted when the organization is eager to move forward. This avoids delayed deliveries and the considerable cost of re-work.
- Security products can be implemented in a working automation practice to shorten feedback loops. As soon as code gets checked-in or built, it is evaluated for security, and if it is assessed as a challenge, a warning can be issued to the developer.

## Risk reduction

Risk mitigation is critical for any company, adopting DevOps makes this even more so. As change frequency and volume are driven up, focusing on potential risks throughout the development and release processes is a necessity.

Implementing any kind of automation will help drive changes quicker and more efficiently.

But implementing controls to prevent a release of the wrong/bad/challenged code will stop you from breaking your production or customer sites too easily.

## Easily enforced compliance

Policies and guidelines should be in place to enforce compliance and follow industry best practices, organizational rules, and regulations.

Any deviations from these policies should be perceived as a security breach and require action.

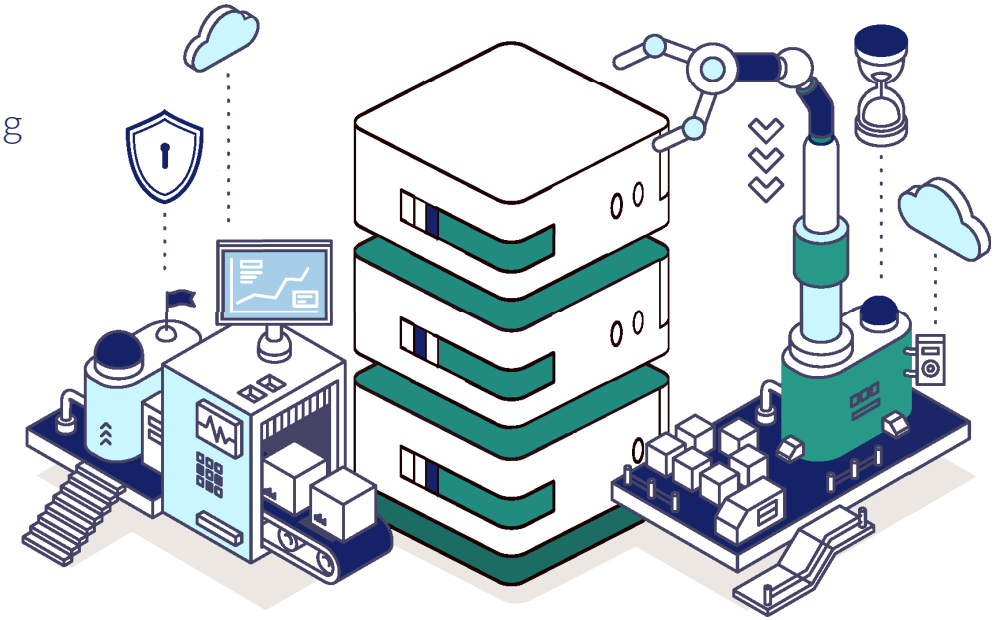
Compliance processes in DevSecOps will require a corrective action to be taken to manage deviations from security or rules.

This can be done through security/policy alerts, notifications, or stopping of the automation line.

# A challenge called The Database

Databases are a challenge whenever we look at Dev, Sec, or Ops.
They require specialized tooling to follow industry development best practices such as database source control, baselining, and labeling changes. Databases also require different methods when adopting operational processes – simply automating database releases can create significant damage, as database configuration drifts are a big risk, and overriding out-of-process changes can destabilize or break releases and environments. So it is not a surprise to learn that databases need special attention when considering the security aspects of DevSecOps.

Below is a list of things to plan for when implementing security safeguards in database DevOps processes. These considerations should be even more important when implementing automation of any kind because the speed and frequency of change will prevent using any type of manual process that is not easily triggered.



# 1 Determine and control roles and responsibilities

• What role does a person play in the organization? Are they a release manager, project admin, DBA, or developer? Each role will require different access and activity rights.

• Which environment can a person introduce changes to? Development? Integration? QA and preproduction? Who will have the credentials to log in to production and update a new version of the software? Or will this be limited to an automated user to prevent the misuse of credentials?

• How are users managed regarding domain logins, database login credentials, and database permissions? Everything needs to fit into one easy-to-manage process so that no one will use another's credentials.

• What type of DB objects, schema, database code, or metadata should someone be able to change? For example, a DBA might be able to introduce table structural changes, while a developer can change database procedures and functions, and a business user can change metadata that determines the behaviour of an application.

# 2 Manage audit & compliance

• Reporting any change introduced to the DB is crucial. Tracking the type of change, who made the change, when, and why, whether through direct database access or via an automated process, will assist audits, compliance, and root cause analysis. Finding out why a performance issue happens during production is much easier if you know that a version was updated or a patch was applied, and what database object was affected.

• Reporting unauthorized attempts to change the DB, either by potential hacking or by an honest mistake, will enable the organization to identify rogue activities, potential misconduct, and in many cases, broken processes that need to be fixed.

# 3 Manage organizational policies

• As soon as any type of automation is introduced, defining and enforcing a clear policy of what an update may or may not include either becomes a bottleneck, if you have to wait for someone to review code changes, or a potentially quick and efficient feedback loop, if implemented as an automated step with DevSecOps.

• Policy should be able to deal with:
• Qualifying changes – should a developer be able to delete a database table? Obviously, the answer depends on whether this is done in development or in staging and production.
• Was the change written in a secure manner? Can it be easily hacked?
• Is the change violating best coding practices? This might not be a clear security issue, but poor coding might bring your system down, so it needs to be evaluated as well.
• What environments should the policy apply to? Truncating content of a table might be fine for development or QA environments but have catastrophic results in production.
• Timing – rebuilding an index is a straightforward change, but doing this in production, mid-day, on a critical table will easily spell downtime.

• Who has the authority to approve deviations from policy? As part of code review processes, or as part of approving or rejecting planned releases that are not meeting pre-defined criteria?

# 4 Enforce processes

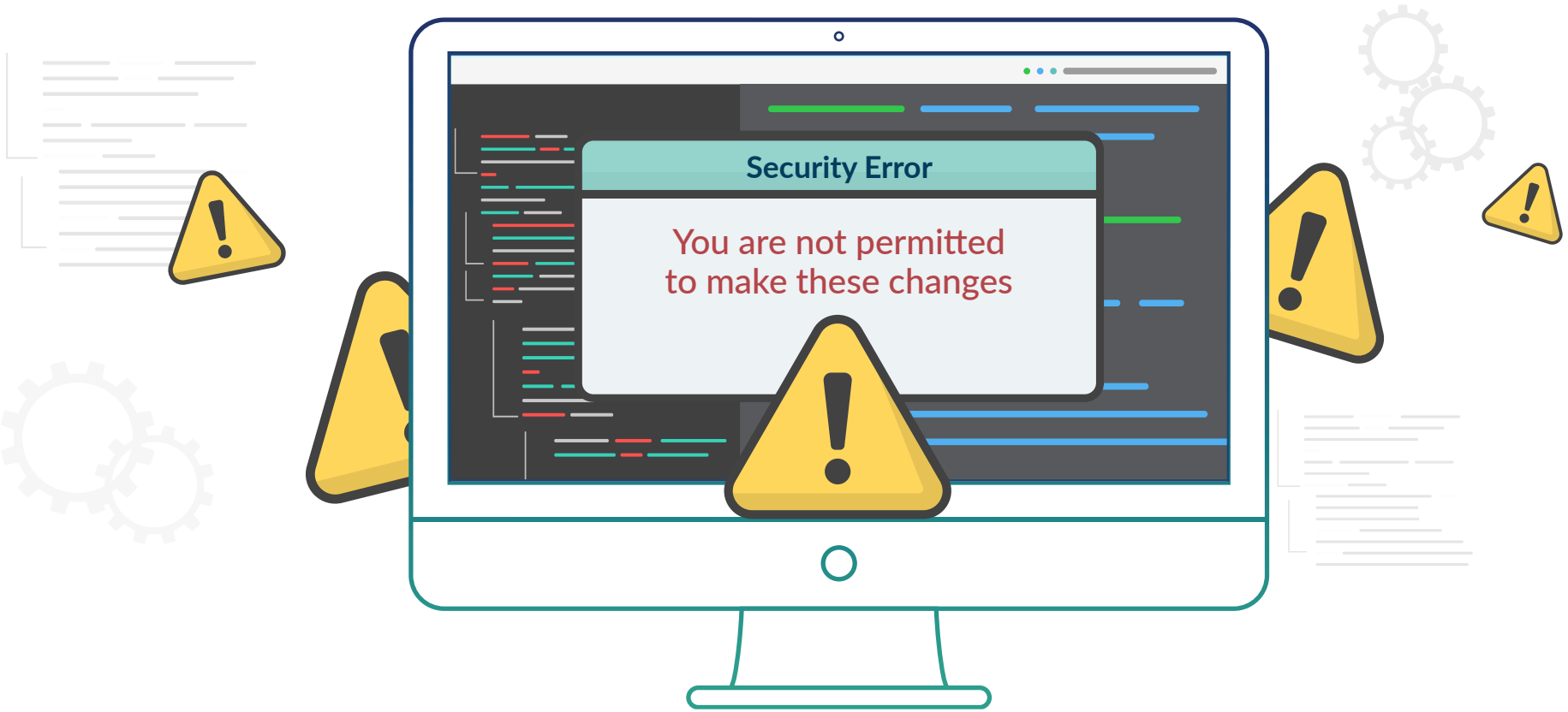
• Preventing broken or out-of-process changes is critical for a stable release process as part of DevOps because any process will fail if one can easily bypass it.

• Roles and permissions need to be enforced to make sure the process is repeatable and everyone is doing only what they should.

• Enforcing policies to ensure that they are automatically checked will make certain that:
  i. They don't become recommendations that are never followed
  ii. They don't become bottlenecks requiring manual review

# Conclusion

Implementing security and compliance should be done as early as possible in any automated or DevOps process. DevSecOps will enable you to make sure you can provide a secure application to your customers while assuring quickest time to market and a well-ironed process. Issues will either be fleshed out early or efficiently reported back to development with a short feedback loop. The overall process will become more secure, effective, and less costly due to the prevention of reworked solutions and reduced downtime.



# About DBmaestro

DBmaestro is a leading DevOps for database solution provider. Our flagship product, DBmaestro DevOps Suite, introduces DevOps and automation best practices to databases for the enterprise, dramatically simplifying, accelerating, and improving release processes, while modernizing database development via release pipelines, long enjoyed elsewhere in the industry.

We provide both database source control and database release automation capabilities across the board for developers, DBAs, security, and operations in multi-database enterprise environments.

## For more information please visit us:

DBmaestro
DevOps for Database