What is database continuous integration?

Database continuous integration (CI) is the rapid integration of database schema and logic changes into application development efforts and to provide immediate feedback to developers on any issues that arise. As a further evolution, database continuous delivery (CD)seeks to produce releasable database code in short, incremental cycles. The goal of these practices is to reduce time to market and create a steady stream of end user value with frequent, high-quality software releases.

To deliver new software experiences more quickly to market, companies have spent the last decade trying to modernize the way they build and deliver software. In their efforts to modernize, many tools have focused on bringing CI and CD to application code. However, the same has not happened for database code (see Figure 1). Many organizations still rely on a shared service database team that manually reviews and deploys DB code changes. Given that the end user experience is incomplete without the full software stack, which includes both the application *and* the database, there has been growing interest and demand in continuous integration and delivery tools for the database as well.

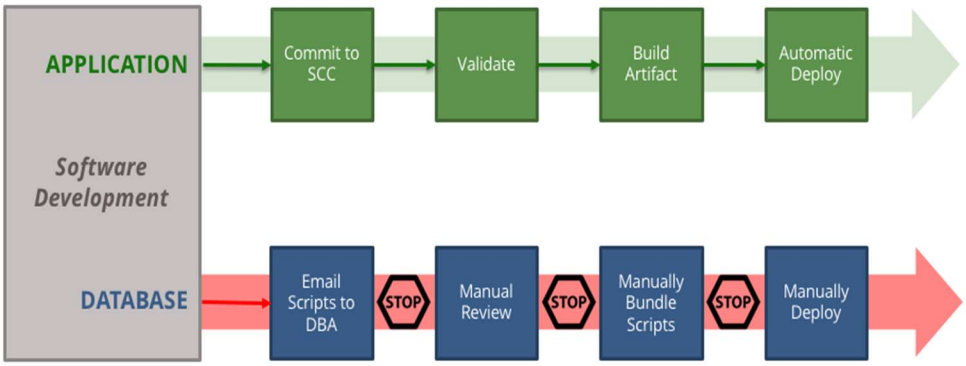

Figure 1. Continuous Integration: Before Database Deployment Automaton

How do database continuous integration tools impact database deployments?

As stated before, **the aim of database CI tools is to bring the same integration and deployment best practices to the database and enable SQL code to flow through the software release pipeline, synchronized with application code** (see Figure 2). By aligning database deployments with application deployments, teams should see a much better return on their investments in the tooling and process updates. This should also help teams bring new innovations to market faster and with higher quality.

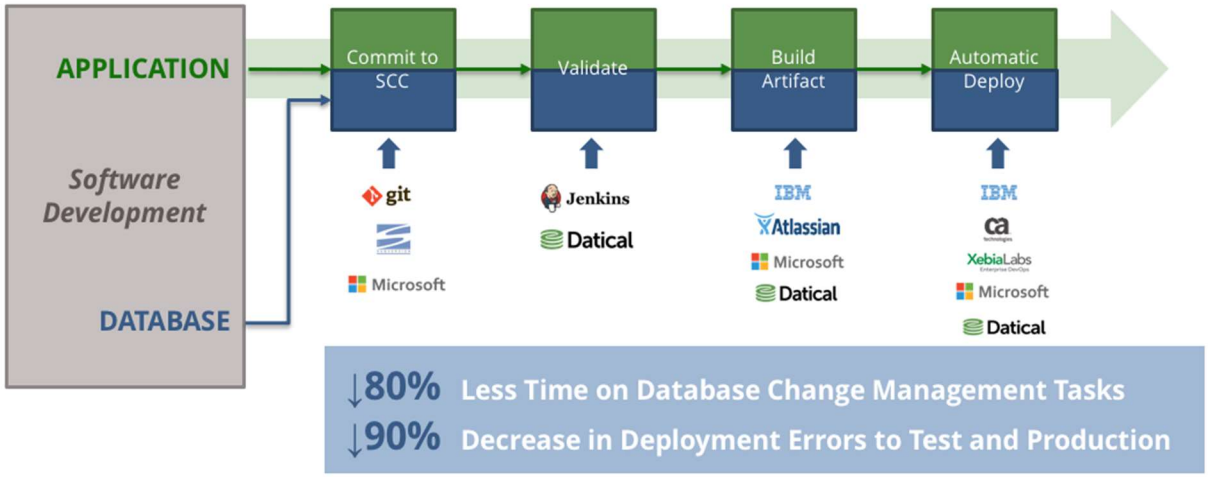

Figure 2. Continuous Integration: After Database Deployment Automation

Database Deployment and Database Continuous Integration Best Practices

Many of the best practices that apply to application CI and delivery tools readily apply to the database as well. These best practices include:

1. Tracking Database Code Changes:

Database code changes should be tracked in the same source or version control system as application code. Database code should not be treated separately or tracked in an entirely different system. A separate system for database code leads to duplicated effort, lack of visibility, confusion, and errors. Application and database code will also begin to drift on a separate system and get misaligned. Automated DB deployment tools like Datical allow teams to push database code into the source or version control solution that is already in place for application code.

2. Automated Database Code Validation and Feedback:

Once application code is checked in, a series of automated test are immediately triggered. These tests will assess if there are any issues in the code that warrant re-work. The same isn't the case for database code. One of the biggest challenges in accelerating database deployments is getting rid of the manual SQL code review that DBAs must perform.

This tedious manual effort can and should be largely eliminated by intelligent automation. That way, developers can get immediate feedback on SQL code – just as they do with application code – and avoid a long wait state in which they move on to a different task. This inefficiency in feedback causes large delays in database deployments and contributes to poor quality. It also forces developers to context switch to make fixes on SQL code changes that they wrote days or weeks ago.

To properly automate the validation of database code, a database deployment automation solution must have an object-model of the proposed SQL code change. Otherwise, functional rules, such as ensuring that all tables having a primary key or unique constraint, cannot easily be validated. This means a DBA will need to give the code manual attention – which also means developers don't get immediate feedback on changes.

Datical's Dynamic Rules Engine is unique in that it is an object-based rules engine that can be easily extended. The Dynamic Rules Engine gets rid of much of the tedious DBA review otherwise required.

Be wary to stay away from simple, regular-expression based rules systems. It's impossible for these systems to functionally validate the common organizational rules and standards that DBAs end up spending much of their time and energy on, instead of making progress on more critical value-add projects such as performance tuning, data architecture, high availability strategy, system upgrade planning and more.

3. Packaging Database Code:

An important DevOps mantra is to "build once, deploy often." Effectively all continuous integration tools allow application code to be built into an immutable package for consistent, repeatable, and predictable downstream deployment. Should anything go wrong, its errors can immediately be traced either to the application code or the environment when working with an immutable artifact.

Database continuous integration tools bring this same advancement to database code. As an example, Datical's Database Code Packager creates an immutable, idempotent artifact from validated database code so automated DB deployments can enjoy the same consistency, repeatability, and predictability as application code releases.

4. Providing Visibility into Database State

Another key DevOps tenant is to "amplify feedback." Continuous integration solutions focus on providing visibility and feedback readily and immediately. As such, a database continuous integration solution should have an accessible web interface. This interface should allow all stakeholders to quickly understand the status of every database. Furthermore, it's important for the solution to integrate with ticketing systems such as JIRA and TFS, and to support parallel development strategies commonly found across enterprise development teams.

DB deployment automation tools like Datical provide a web interface and have labelling systems that integrate with ticketing solutions to provide necessary visibility and feedback to appropriate stakeholders. With Datical, all DB code changes can be traced back to source code control. This quickly links the changes to the business value that they are meant to deliver.

Continuous Integration and Database Deployment Automation Tools

There are four categories of database deployment tools you should have in your automation framework to achieve database continuous integration:

Database Development Tools

When you think of database deployment tools, it's unlikely you'll consider application development tools as part of that category. However, all your tooling should work together to achieve database continuous integration. Just as application development tools have grown to integrate with CI solutions, the same is true for database development tools as well. Quest Software's Toad has some great features for supporting Agile database development. The first is its Team Coding functionality. Databases are different than the application and have state. Thus, having a database instance to develop against as part of a team is key. Toad's Team Coding allows users to leverage existing Source Code Control (like Git, Subversion, others) and a live database to support database developers. Not only can they check in and out their SQL scripts, but they can do so with the database objects like tables and stored procedures, as well.

Database Release Automation Tools

Beyond an appropriate development tool, database CI requires a DB release automation solution. Database release automation tools like Datical deliver automated validation, build, test, and deployment of database changes. These core capabilities ensure that any database code pushed to source code control is pulled out into a CI process that can provide developers with near immediate feedback. With a database release automation solution, software teams can consistently deliver a continuous stream of value to end users without getting slowed down by database deployments.

Application Release Automation Tools

As organizations add new features and enhance existing software, the number of components and the complexity of the software stack have only grown. As new trends emerge that allow functional isolation and which avoid single points of failure, orchestrating and aligning the release of all the necessary components requires application release automation (ARA). Tools like CA's Automic, IBM's UrbanCode Deploy, Serena's Deployment Automation from MicroFocus, and XebiaLab's XL Deploy integrate with

Database Release Automation solutions like Datical to enable continuous integration for the full software stack.

Test Data Management Tools

A key trend in software development is test-driven development, in which the CI process is enhanced with production-quality test data. It's common for applications to change functionality based on the data stored in the database. Consequently, providing data that mirrors (or matches) production data is essential to a database continuous integration process that can generate high-quality test output.

DevOps Tools like CA's TDM, Delphix Data Virtualization and IBM's Optim, populate test databases with data. Some solutions, such as Delphix, allow users to request masked self-service copies of production databases for integrated testing environments.

Continuous integration and continuous delivery are best practices for accelerating the speed and quality of application code changes. Similarly, DB continuous integration is important in accelerating the database release process while reducing risk. By including a database release automation solution and adding database CI to your existing application delivery toolchain and process, you can begin to increase the pace and quality at which the entire software stack – including both the application and database – can be delivered to market.