INTRODUCTION TO DATABASE DEVOPS

Database DevOps is an emerging area of the DevOps methodology. Let's take a look at database management and what happens when you apply DevOps concepts.

(This article is part of our DevOps Guide. Use the right-hand menu to go deeper into individual practices and concepts.)

DevOps for databases?

DevOps is (officially) the preferred software development lifecycle (SDLC) framework for application development in recent years. Continuous operations and automation have emerged as a key characteristic of the DevOps philosophy as practiced in successful DevOps environments.

Yet, the principles of continuity and automation have not entirely encompassed all aspects of software applications.

The ongoing management, change, updates, development, and processing of database (code) has emerged as a bottleneck for many DevOps organizations. This has forced engineers to invest countless hours on database development rework that support continuous release cycles as expected for a streamlined DevOps SDLC pipeline.

Since database changes and development is considered as one of the riskiest and slowest processes of the SDLC, applying the DevOps principles of continuous operations and automation specifically for database code development is seen as a potential solution to the database problem. According to a recent research report:

- Over 90% of application stakeholders work toward accelerating database deployment and

management procedures.
- More than half of all application changes further require modifications to the corresponding database code.

Database challenges in DevOps

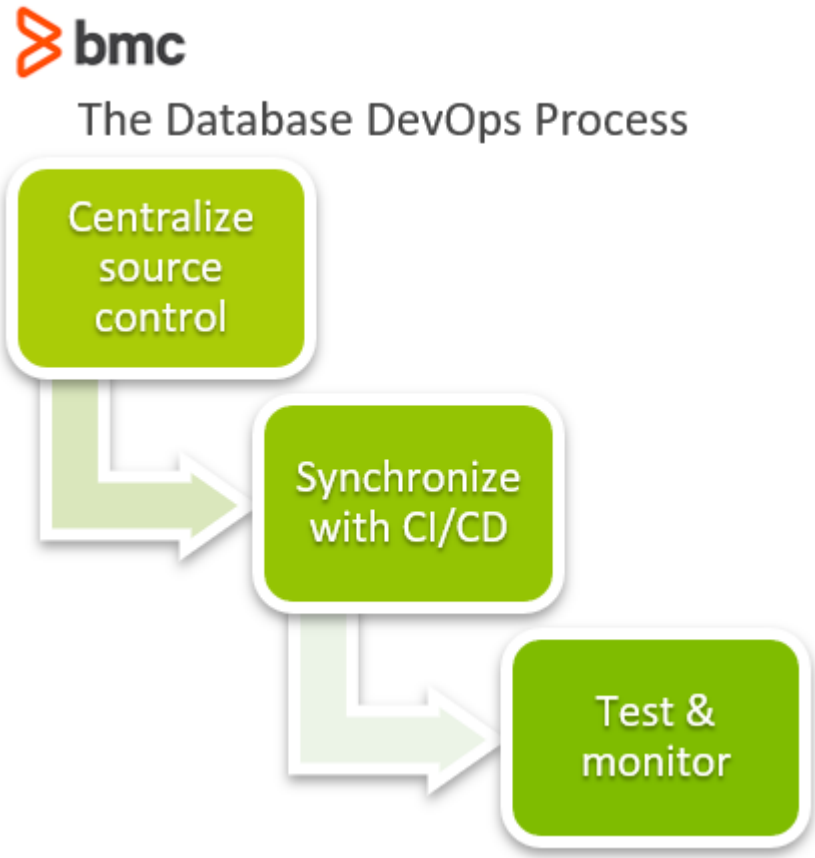
Before we discuss how database DevOps can make the DevOps SDLC pipeline efficient, let's discuss the database-specific challenges facing DevOps organizations:

- **Manual changes.** Traditional database management follows manual processes such as code reviews and approval—all of which hold up the release cycle.
- **Data provisioning.** Due to security and regulatory limitations, data from production is often not available to test early application builds. The data is therefore processed and encrypted to address the necessary regulatory requirements.
- **CI/CD for database.** Data persistence cannot be maintained in the same way as code persistence is managed for a Continuous Integration/Continuous Deployment (CI/CD) pipeline. Continuous integration and deployment of new database versions have to respect the common structure and schema of databases—which is precisely why manual intervention becomes necessary.
- **Integration challenges.** The sheer variety of tooling and architectural variations can make it difficult for database systems to work coherently. The lack of standardization means that a DevOps teams cannot entirely follow continuous and automated infrastructure operations for database system provisioning and management.

And then there's a bigger, harder-to-tackle challenge: **insufficient DevOps attention.**

Many real-world DevOps implementations have failed to integrate database and application development process into a unified and holistic SDLC framework policy. Database management processes have continued the traditional route, and the increasing scale of database changes has made it difficult for engineers to standardize and coordinate database development efforts with the rest of application development.

(Watch these challenges grow as data storage increases.)



What's Database DevOps? A process

Now, let's look at the main tasks involved in Database DevOps, which in fact make the process similar to adopting the DevOps framework for application code:

1. Centralize source control

Use a centralized version control system where all of the database code is stored, merged, and modified. Storing static data, scripts, and configuration files all within a unified source control system makes it easy to roll back changes and synchronize database code changes with the application code development following a CI/CD approach.

(Learn more about CI/CD.)

2. Synchronize with CI/CD

Automate build operations that run alongside application releases. This makes it easy to coordinate the application and database code deployment process. The database code is tested and updated at the same time a new software build integration takes place, according to the underlying database dependencies.

3. Test & monitor

The CI/CD approach with a centralized version control system for both database and application code makes it easier to identify database problems every time a new build is checked in and compiled to the repository.

- Developers can identify and address problems earlier during the SDLC lifecycle that follows continuous deployment and release cycles of a fast-paced DevOps framework.
- Conversely, traditional database management operations require slow and manual review, testing, and approval processes for the corresponding database code.

Database DevOps best practices

Additional best practices consistent with the DevOps framework:

- **Adopt the DevOps and Agile principles** of writing small, incremental, and frequent changes to the database code. Small changes are easier to revert and identify potential bugs early during the SDLC process.
- **At every incremental change, monitor and manage dependencies.** Follow a microservices-based approach for database development.
- **Adopt the fast feedback loop** similar to the application code development process. However, critical feedback may be hidden within the deluge of log metrics data and alerts generated at every node of the network.
- **Track every change made to the database code.** Test early and often, prioritizing metrics performance based on business impact and user experience.
- **Set up the testing environment to replicate real-world use cases.** Establish a production environment to stage tests that ensure dependability of the database.
- **Automate as much as possible.** Identify repetitive and predictable database management tasks and write scripts that update the database code when a new build is compiled at the Continuous Integration server.

Finally, note that every DevOps implementation is unique to the organization adopting the framework. Database DevOps can conceptually take several guidelines from the application code DevOps playbook, and integrate database code development and management along with the application code for similar SDLC performance and efficiency gains.

Related reading

- BMC DevOps Blog
- Database Automation Explained: Concepts & Best Practices
- An Introduction to Database Reliability
- What Is a Database Reliability Engineer (DBRE)?
- CAP Theorem for Databases: Consistency, Availability & Partition Tolerance
- Database Administrator (DBA) Roles & Responsibilities in The Big Data Age