harness

Integrating Database DevOps:

The Path to Seamless CI/CD and Guaranteed Data Integrity

This comprehensive guide explores four key use cases that demonstrate how to integrate data and databases into your CI/CD pipeline, ensuring seamless operations and enhanced reliability.

Integrating the database directly into your CI/CD pipeline is no longer optional—it is a transformative approach to modern software delivery. This is the essence of **Database DevOps,** which solves the critical challenge of coordinating application and database changes.

By treating database changes as code and including them in your continuous delivery workflow, you can:

- **Bridge the Deployment Gap:** Eliminate the manual, error-prone separation between application and database deployments, ensuring they are always synchronized.
- **Significantly Minimize Downtime:** Implement schema and data migrations with precision and speed, reducing the risk of outages and service interruptions.
- **Improve Reliability and Consistency:** Guarantee that the application is always running against a compatible database schema, leading to fewer failed deployments and a higher level of confidence in every release.

Ultimately, a unified, data-leveraged CI/CD pipeline accelerates your delivery cycles, fosters a culture of continuous improvement, and ensures the reliability and efficiency of your entire Software Delivery Lifecycle (SDLC).

Let's dive into the four key use cases that demonstrate how to integrate databases into your CI/CD pipeline.

harness

Use Case 1: Migrating Data Between Environments

One of the biggest challenges in practicing proper CI/CD is ensuring that testing environments perfectly mirror production, particularly with representative test data. A mismatch between environments is a major cause of unexpected outages and incidents in live systems that should have been caught in staging.

To solve this challenge, a powerful strategy is to leverage the **CI/CD** pipeline to replicate production data directly into a dedicated test environment. This isn't just copying data; it's creating a testing ground that truly reflects the complexities of your production system, allowing you to identify and resolve costly issues before they impact users.

The necessity for a realistic testing ground extends to the critical issue of obtaining and utilizing test data that accurately mirrors real-world scenarios. When replicating production data, the test environment must strictly adhere to the same rigorous security safeguards as your production environment. Security should be an integral part of the design, as failing to meet this standard introduces unacceptable risks and compromises the integrity of your testing process.

With this rigorous approach in place, leveraging production data offers an invaluable advantage that delivers several significant benefits:

- **Uncover Hidden Patterns and Issues:** Test data, by its very nature, is often curated and simplified. It may not capture the intricate data relationships, edge cases, or historical patterns that exist in a live production database. Replicating production data allows you to expose these subtle yet critical issues that might otherwise remain dormant until deployment, leading to unforeseen failures.
- **Realistic Performance Testing:** The performance characteristics of your application can vary significantly depending on the volume and complexity of the data it interacts with. Testing with production data enables you to conduct more accurate and reliable performance benchmarks, identifying bottlenecks and scalability challenges before they impact your users.
- **Enhanced Confidence in Deployments:** When you can validate your application's behavior against actual production data in a controlled environment, the level of confidence in your deployments dramatically increases. This significantly reduces the risk of unexpected failures and rollbacks, leading to smoother and more predictable release cycles.

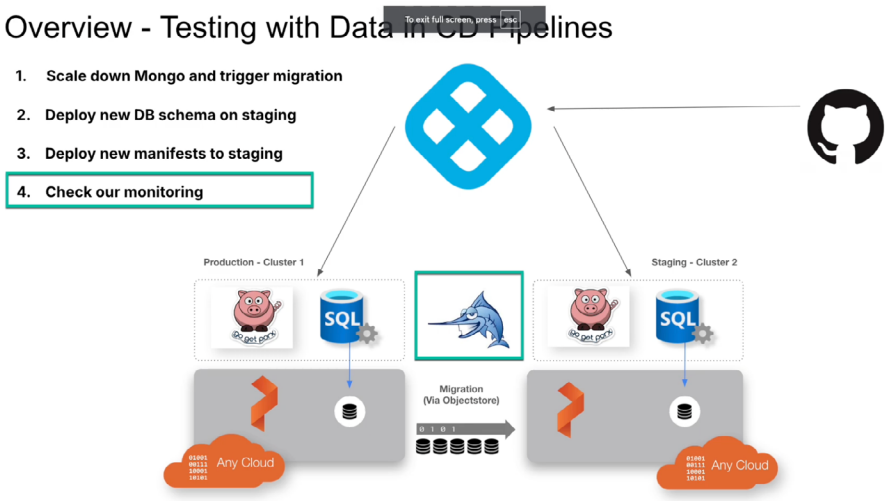
- **Validation of Data Migration Strategies:** For organizations undertaking complex data migrations or database upgrades, replicating production data into a test environment provides a safe and effective sandbox to validate their migration strategies. This includes verifying data integrity, schema compatibility, and the performance of migration scripts, all without impacting the live production system.

This essential data replication process can be accomplished through several common methods, ensuring your test environments accurately reflect production. The most relevant options include:

- **Read Replica Promotion:** Setting up a read replica of the production database, allowing it to sync completely, and then promoting it to an independent master database within the test environment.
- **Backup and Restore:** Alternately, you can take a full backup of the production database and restore it to the pre-production environment.

A specialized and powerful option exists if your database is running on Kubernetes:

- **Portworx Migration:** In a Kubernetes environment, you can copy data between clusters using a **Portworx migration**. Portworx provides the robust capabilities necessary to create and manage persistent storage for containerized applications, making it an ideal tool for facilitating the secure and reliable replication of production data volumes into testing environments. By leveraging Portworx, development and operations teams can ensure that their pre-production environments are not just approximations, but accurate reflections of the production landscape, ultimately leading to more stable, secure, and performant applications.



harness

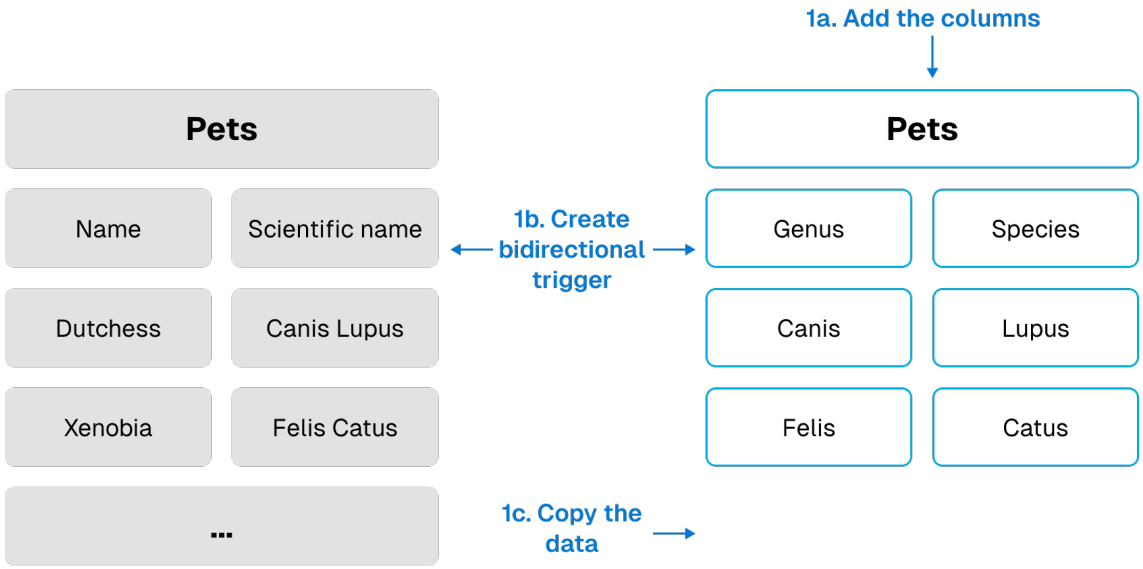
Use Case 2: Database Schema Changes with Zero Downtime

Introducing changes to your database schema without causing downtime is a **critical requirement** for many applications, especially those operating 24/7 with high availability demands. This process, often referred to as a "zero-downtime schema migration," is crucial for maintaining a seamless user experience and preventing service interruptions.

A robust strategy for achieving this involves a two-step approach that ensures both old and new versions of an application can interact with the database simultaneously during the transition:

1. **Adding New Columns with Backward Compatibility:** The first step involves introducing new columns (e.g., adding city and state columns to capture more granular location data) while keeping the existing, older column (e.g., a general location column) intact. This dual-column strategy ensures backward compatibility, allowing older versions of the application that reference the original column to continue functioning without errors. This permits a phased rollout where different application versions can coexist during the transition.
2. **Setting Up Database Triggers for Data Synchronization:** To ensure data integrity and consistency, database triggers are implemented. These automated actions, executed in response to specific events (e.g., INSERT, UPDATE, DELETE) on a table, are configured to synchronize data between the old and new schema columns in real-time. For example, an update to the new city/state columns could automatically update the old location column, and conversely, an update to the old location column could populate the new city/state columns. This real-time synchronization is vital for maintaining a consistent view of the data across all application versions throughout the migration.

The combination of dual columns and triggers ensures that both the old and new versions of the application can work against the updated schema, providing a smooth transition without any downtime. The intelligent use of triggers is instrumental in enforcing referential data integrity and consistency, ensuring that the information stored in the database remains accurate and reliable throughout the schema evolution process. This effective strategy is a cornerstone of modern application development, supporting continuous delivery and deployment in a highly available environment.



Use Case 3: Deploying Application Updates

Deploying a new version of an application that depends on recent database changes can be challenging. This section effectively addresses these complexities by leveraging two powerful tools from the Harness platform. Harness CD to deploy the new version of the application and Harness Database DevOps to deploy changes to the database:

- **Harness CD (Continuous Delivery):** This tool is utilized for the streamlined deployment of the new application version. Harness CD automates the entire software delivery pipeline, from build to deployment, ensuring consistency, speed, and reliability in application updates. Its robust capabilities help to manage various deployment strategies, rollbacks, and environment promotions, making application deployments efficient and less prone to manual errors.
- **Harness Database DevOps:** This solution is employed to manage and deploy changes to the database. Harness Database DevOps brings DevOps principles to database management, allowing for version control, automated schema migrations, and synchronized deployments of database changes. It provides a secure and controlled way to evolve database schemas alongside application code.

By integrating database schema changes and application updates within the same pipeline, you can ensure that both are deployed in a coordinated manner. Similarly, if either needs to roll back, they can roll back together. This approach minimizes the risk of compatibility issues and ensures your application remains stable and functional. Deploying a new version of an application, especially when it relies on recent database changes, presents a significant challenge in modern software development. Ensuring seamless integration and preventing compatibility issues between application code and database schemas is paramount for maintaining application stability and functionality.

The core innovation in this approach lies in the integration of database schema changes and application updates within the same unified pipeline. This coordinated deployment strategy offers several critical advantages:

- **Minimized Risk of Compatibility Issues:** By deploying application and database changes together, the likelihood of an application failing due to an incompatible database schema is drastically reduced. The pipeline ensures that the application always interacts with a database schema it expects.
- **Guaranteed Consistency:** Both application and database components are updated in a synchronized manner, leading to a consistent state across the entire system. This eliminates the “drift” that can occur when these components are deployed independently.
- **Simplified Rollbacks:** A significant benefit of this integrated approach is the ability to perform coordinated rollbacks. If any issues arise post-deployment, both the application and the corresponding database changes can be rolled back together to a previous stable state. This ensures that the system remains functional and avoids partial rollbacks that could leave the application in an inconsistent or broken state. This capability is crucial for rapid recovery from unforeseen problems, minimizing downtime and business impact.
- **Enhanced Stability and Functionality:** The combined effect of these benefits is a more stable and functional application. The automated, coordinated deployment process reduces manual errors, accelerates delivery cycles, and ultimately improves the end-user experience by ensuring that the application is always running on a compatible and optimized database.

By treating database changes as an integral part of the continuous delivery pipeline, organizations can achieve true end-to-end automation, fostering a more reliable, efficient, and robust software delivery process. This holistic approach to deployment not only minimizes risks but also accelerates innovation by allowing teams to confidently and frequently release new features and updates.

Data Governance and Integrity

For industries such as HealthTech and Finance, where Data Governance is a primary concern, the risks associated with data loss and corruption are critical factors. A robust CI/CD pipeline must address these concerns by prioritizing data integrity alongside deployment speed and efficiency.

Harness Database DevOps and CD tackle these challenges directly by providing a framework that ensures safe and compliant data operations:

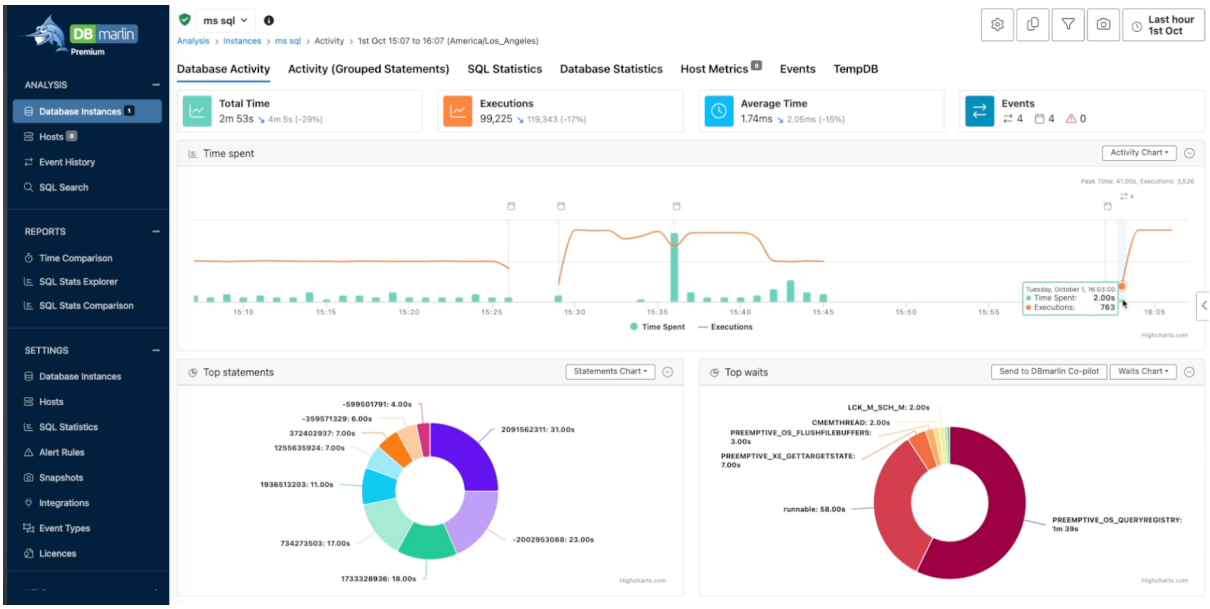
- **Preventing Data Loss and Corruption:** By automating database changes and treating them as version-controlled artifacts, the pipeline ensures that every modification is tracked, reviewed, and tested before reaching production. This structured approach significantly reduces the human errors that often lead to data corruption or accidental loss.
- **Ensuring Data Integrity:** The ability to synchronize application updates with database schema changes guarantees that the application always interacts with a compatible database structure. Furthermore, the coordinated rollback capabilities mean that if an error occurs, the entire system—both code and data schema—reverts to a known good state, preserving the integrity of the data and the application state.

This approach ensures that organizations can maintain strict governance standards, protecting sensitive data while still benefiting from the velocity of modern DevOps practices.

Use Case 4: Monitoring Database Health

Monitoring the health of your database throughout the Continuous Integration and Continuous Delivery (CI/CD) process is crucial for identifying and addressing performance issues. A common scenario is a schema migration that unintentionally refactors a query, causing it to join two distinct database tables on unindexed columns, which leads to performance degradation.1

Solutions like **DBmarlin**—a database monitoring solution—provide visibility into changes applied by tools such as Harness DB DevOps and Harness CD. DBmarlin intelligently correlates deployment changes with real-time monitoring data. Other comprehensive database and Application Performance Management (APM) tools, such as **Datadog** or **New Relic**, also offer similar integration capabilities to track performance alongside application and database deployments.



This correlation capability empowers users to:

- **Detect Performance Issues with Precision:** Accurately identify and pinpoint any negative performance impacts, whether they stem from newly introduced database schema changes or recent application updates. This granular detection allows for swift diagnosis and remediation.
- **Optimize Queries for Peak Performance:** Leverage intelligent optimization guidance to iteratively improve query performance. This guidance helps developers and database administrators refine problematic queries, leading to more efficient database interactions
- **Compare Performance Across Deployments:** Generate detailed run-time comparison reports. These reports provide a clear, side-by-side analysis of the impact of recent changes on query execution times, enabling data-driven decisions about the health and efficiency of the database.

After an issue like this is detected, the pipeline can deploy updated indexes to fix the performance bottleneck. Continuously monitoring the health of your database is critical for the early identification and proactive resolution of potential performance bottlenecks and issues throughout the entire CI/CD pipeline.

Learn More About CI/CD for Databases

By adopting a comprehensive Database DevOps approach and integrating powerful tools like Harness Database DevOps and Harness CD, your organization can create a highly efficient pipeline. This ensures smooth transitions, minimizes downtime, and maintains optimal performance, significantly improving the reliability and efficiency of your development and deployment processes. If you are interested in learning more about how Harness can help you manage database changes within your CI/CD pipeline, please **reach out to Harness.**

Find more insights about database DevOps:

- **Database DevOps Overview**
- **Database DevOps: Lessons Learned from Migration Hell**
- **AI in Database DevOps: From Manual Bottlenecks to Autonomous Change Management**

harness

The AI-Native Software Delivery Platform™

Follow us on

/harnessio

/harnessinc

Contact us on

www.harness.io