WHITEPAPER

Metrics to modernize database change management

redgate

Contents

Why do software delivery metrics matter?

Software delivery metrics matter. Their impact is felt not only by software delivery teams, but also right down to a company's bottom line.

DORA's research program has long-established software delivery metrics that are adopted by software teams to prioritize continuous improvement. Research shows that top performers are able to[1]:

- Deploy on demand
- Have a less-than-one-day lead time for releasing changes
- Run with a change failure rate of 5%
- Operate with a failed deployment recovery time of less than one hour

And those top performers see resulting benefits, including increased performance and job satisfaction for software delivery teams in addition to stronger customer satisfaction.

Understanding context is essential

But context is important. For some organizations and for some applications, a deployment on demand schedule may not be the perfect fit. For example, the requirements for developing a mobile application are going to be very different to working on a mainframe system.

Organizations who understand the context and apply the right baselines and metrics to fit will gain the most.

[1] DORA Research: 2023, Overview

The four metrics that matter

DORA defines four key metrics that provide an effective way to measure the outcomes of the software delivery process[2]:

1. Change lead time
2. Deployment frequency
3. Change fail percentage
4. Failed deployment recovery time

Change lead time

Change lead time is the time it takes from the point of committing code to code running in production. It helps organizations understand the efficiency of the development pipeline. It can also point to how effective an organization's CI/CD processes are if they are implemented.

Deployment frequency

This metric measures how often new code is deployed to production. The more agile the team, the more frequently they will be able to deploy new code. If a team is able to deliver new features, enhancements and fix bugs frequently (in a little and often approach), they are not only releasing continuous value to customers, but also reducing risk. A further benefit is that the more frequently deployments are released, the more important robust testing becomes, which in turn drives up quality.

Change fail percentage

Change fail percentage, sometimes known as change failure rate (CFR), represents the percentage of deployments that result in a failure in production. This is a key metric for indicating stability and the quality of the code being deployed. It is also a high priority metric – velocity and frequency of deployments becomes irrelevant if releases are not stable. High change failure rates also mean that you're spending more time fixing changes instead of getting them into Production and out to market.

[2] DORA Guides, DORA Metrics: Four Keys

Failed deployment recovery time

This metric is sometimes known as mean time to recovery (MTTR) and it reports on the average time it takes to recover from failure in the production environment. A low MTTR signals greater resilience and how successfully teams are able to address issues and minimize the costly effects of downtime, while maintaining service availability.

Influencing investment decisions

These metrics are important because ultimately, they influence investment decisions. By tracking and optimizing the four keys for software delivery performance, organizations can release changes to market with speed and reliability. And by coupling speed and stability in the software lifecycle, organizations are in a better position to scale, invest and innovate.

But what about the database?

Thousands of organizations use software pipelines and adopt organization performance metrics to optimize the release of application changes to market. But what about the database? The requirement to protect business-critical data can be a bottleneck in agile software delivery. The bottleneck can present in different ways, particularly if teams need to take measures to protect sensitive data or are working with accrued technical debt and the complexities that presents. But excluding the database from those same pipelines and processes presents further challenges, exposing risks to application performance, and data security and integrity.

So how do teams unblock bottlenecks in database change management in order to release value to customers faster? The answer lies in a robust database change management process that extends the same practices as are used for application delivery to the database.

Again, when it comes to the performance of that part of the process, context is everything. Applications that operate with a monolithic database at the backend will run with a different set of baselines for the four key metrics, compared with a new mobile application that has a greenfield database behind it.

Taking first steps

The first crucial step is in getting your baseline metrics for the four keys working for the database change management part of your software delivery process.

When you have your baselines in place, you can monitor performance and consider how best to optimize those for greater stability and faster time to market.

Asking the right questions

Baselining metrics for the database change management part of your process also enables you and your teams to ask crucial questions, both to develop your database change management strategy and fine-tune performance.

Depending on how your metrics are performing, questions you might ask include:

1. What is causing our change failure rate?
2. What is the cost of change failures in the application tier vs the database tier?
3. Why does it take so long to restore the service in line with your SLAs?
4. Once we've restored the service, how much data is acceptable to lose? Can data fixes be more easily deployed alongside code fixes for a less complex and faster service restoration? Conversely, would we rather have a zero recovery point, but know it will take us potentially as long as 8 hours to get there?
5. Where are our backups? How can these improve our recovery point objectives?

Communication is everything

Once you have your metrics and you're asking the right questions, communication is key to unlocking the next level: data-driven decision making that enabling teams to self-diagnose what needs to be improved.

- The metrics provide clarity and transparency – ensure the right team members have access to the metrics that matter for their roles. This builds trust.
- By agreeing regular review points for the metrics, teams can proactively identify bottlenecks and areas for optimization.
- Open discussion of metrics breaks down silos and fosters a more collaborative approach to problem-solving. This sort of open discussion often works best when the team culture fosters a no-blame approach and a willingness to learn from mistakes.

Aligning metrics with business goals

It's easy to get into the weeds of the data, but what really counts is applying those software delivery performance metrics with business goals. Common questions to ask include:

1. Quality code matters to the customer experience – what do we need to unblock to deliver more reliable deployments every time?
2. We operate in a highly competitive market – what levers can we pull to reduce our change lead time?
3. Retention of our software delivery teams is important for the scalability of our business – which software delivery metrics will most increase teams' job satisfaction?

Key takeaways

By integrating software delivery performance metrics into your SDLC, including your database change management process, you will be able to drive continuous improvement and ensure continuous release of value to customers.

Actions:

1. Establish the four metrics and get a baseline to work from
2. Ensure a solid understanding of how those metrics align with business goals
3. Monitor performance against those metrics
4. Encourage data-driven decision making, using the metrics to drive optimization

Where Flyway can support monitoring and optimization of software delivery metrics

Change lead time	Use Flyway's flyway migrate –dryRunOutput in your CI pipeline to capture deployment scripts and measure potential lead time automatically. Integrating Flyway with CI/CD pipelines enables teams to track how long it takes to from pull request/commit to deployment.
Deployment frequency	Flyway's script generation removes human error and eliminates time-consuming coding by hand. Efficiency gains help teams to deploy small changes more frequently, while Flyway's robust database comparison technology drives up quality of deployment scripts for release. Again, integrating Flyway with CI/CD pipelines means issues are detected and fixed earlier, meaning teams are in a better position to deploy smaller, iterative changes more frequently.

Change fail percentage	Integrating Flyway with CI/CD pipelines means issues are detected and fixed earlier, helping you to drive down your change fail percentage rate over time.
Mean time to recovery	Using undo -> repair -> migrate with Flyway can help teams meet MTTR thresholds. In the cases where rolling back changes with an undo script is not an option, bringing Flyway into your CI/CD pipelines enables teams to troubleshoot problems earlier and faster, accelerating the deployment of fixes.

Next steps

Discover how your organization can benefit from a more robust database change management process and where software delivery metrics can surface the most business impact.

Get in touch.