Liquibase

WHITE PAPER

Full Stack DevOps Metrics You Must Track

Introduction

To keep up with the constant pace of innovation, organizations have turned to DevOps to modernize their application release process to deliver better experiences faster. The DevOps methodology focuses on connecting development with IT operations—two business functions that have historically operated separately. The idea is that by fostering collaboration and breaking down silos between the two teams, businesses have improved visibility into what needs to be measured to avoid interruptions and reduce downtime throughout the software delivery process.

Aside from the challenge of the left hand commonly not knowing what the right hand is doing, many businesses are plagued with the dilemma of how to measure the performance of their IT departments as a whole—from development to delivery. That's where metrics come in. Without metrics, organizations lack a baseline for performance and may not detect that they are dedicating valuable resources to the wrong things. Taking action on things that have little to no impact on the bottom line translates into missed opportunities and inefficiencies. This paper focuses on the key metrics that provide insight into the health of development and operations teams so that organizations have the visibility to detect—and subsequently address—key inefficiencies.

Problem Statement

The following stages illustrate the process of ushering an application through the software delivery lifecycle: code, build, test, package, release and monitor. Code moves from creation to a quality testing stage designed to provide feedback on errors and prevent downstream outages. Code then moves into pre-production, which should nominally be a dry-run of the push to production. Lastly, once the application is released to production, it progresses into a monitoring stage where performance is tracked and user feedback is collected; the cycle then begins anew with new features or enhancements that must flow from development to production. Metrics can be applied throughout the various stages of the lifecycle to identify, measure, and fix delays and breaks.

The following are five key metrics that can quickly indicate issues in a software delivery process and help identify areas of focus for improving velocity:

Figure 1: The DevOps Tool Chain outlines the stages of the software delivery process and illustrates how an application moves from inception to deployment.

1. Deployment Frequency

This metric tracks the rate at which you make deployments. In a world consumed with instant gratification, customers today have come to expect a constant stream of new enhancements. They also expect their feedback to be heard and to be reflected in new capabilities. Looking at Diagram 1, this metric occurs during the release phase of a deployment. The ability to make code changes easily and quickly translates into improved customer responsiveness which leads to greater customer satisfaction and increases a business' competitive advantage.

By analyzing deployment frequency, a business will realize that smaller, more frequent deployments directly translate into fewer changes and more stability in the environment. Fewer changes means that it is easier to identify issues and resolve them quickly. Less frequent deployments tend to be more complex because of their size, therefore increasing the risk of downtime.

2. Change Volume

It doesn't matter how many changes are made if the changes aren't significant. If a company is deploying every 11 seconds to production but only making minimal or trivial changes, the impact to customers is just as small. Referring to Diagram 1, change volume is a key metric that tracks the quantity of the code that makes it through the pipeline to production, and is a direct measurement of exactly what was released.

While smaller deployments are easier to test and release, it's important to ensure that quantity isn't sacrificed simply to meet a deployment frequency metric. Ultimately, the goal is to bring new capabilities into customer hands in a timely manner and to maintain a competitive advantage. It's a red flag if you discover that the net output (change volume times deployment frequency) is steadily decreasing as software teams work focus on improving deployment frequency.

3. Lead Time

Lead time is the cycle time from when new code starts development to when it's successfully deployed into production. Lead time is a measure of agility. If a company has a long lead time, then changes that need to be made to respond to competitors or customers will likely disappoint because the process is too slow. Even if a company is releasing to production every hour, there is a serious issue if a code change takes six months to make it to production because that effectively means there is a six-month delay in realizing the value from that code change. This makes a company slow and vulnerable as it's unable to respond to competitors or key market trends.

This metric takes a bird's eye view of the entire lifecycle of Diagram 1 as it tracks the process from start to the day the code is pushed to production and available to customers. A long lead time indicates that there are bottlenecks or inefficiencies in the deployment process, such as poor code quality which requires many cycles of rework or

manual review and approval processes that are slow and tedious. Examining the cycle time helps visualize areas in the process that are stalling or manual processes that can be automated to reduce delays.

To measure lead time, companies should employ tools, such as application release automation and issue tracking software, that bring insight and provide visibility into how quickly issues are moving from development to production. A pipeline view can help teams track a release through the software delivery lifecycle so that they can easily pinpoint where things might be getting stuck or might be slowing down.

4. Percentage of Failed Deployments

This metric focuses on the percentage of deployments that have caused an outage, have a negative user reaction, or which are otherwise misaligned with business objectives (for example, if a service is slightly more responsive to end users but an order of magnitude more expensive to operate). In the case of a failed deployment, there is inevitably an impact to the bottom line. This can be indirect, in the form of brand reputation or competitive position, or can be much more direct, as in the case of a service outage or performance issue.

In Diagram 1, percentage of failed deployments come into play at the release and monitoring stages. If changes degrade performance or sacrifice availability to users, then something is dysfunctional and the process needs to be reevaluated. For example, if a new feature is introduced but causes user sign-in to take a lot longer than before, then that new feature has degraded the user experience and will result in customer dissatisfaction.

Analyzing the percentage of failed deployments is crucial to an organization's ability to improve performance. The best way to measure percentage of failed deployments is with a combination of customer feedback and monitoring tools that identify failures. At the core of DevOps is the notion that if quality is built-in from the beginning, this metric should reduce over time. If the failure rate doesn't go down, that is an indication that the DevOps process isn't functioning as it should.

5. Mean Time to Recovery (MTTR)

Murphy's Law states that "anything that can go wrong will go wrong." Something will inevitably malfunction in a deployment, and the key metric here is how long it takes to recover from the issue. Similar to the percentage of failed deployments metric, MTTR is measured during the release and monitoring phases of the software delivery lifecycle illustrated in Diagram 1. The clock starts with an issue and stops when a resolution is delivered to production.

This is where the emphasis on transparency and collaboration in DevOps really shines. As teams learn to work together in the face of adversity, they should be able to cohesively identify, trace, and tackle issues at a faster rate over time. When something does go wrong, a faster recovery time will help avoid customer retention issues and lost revenue. Much like the previous metric of percentage of failed deployments, this metric should decrease as the DevOps process matures.

By relying on testing, monitoring, issue tracking, and source code control tools, teams can trace failures quickly, and coordinate on a timely response to issues. By creating best practices using these tools, issues can be solved faster with subsequent releases.

Conclusion

Software built for consumers has evolved dramatically over the years. To keep pace with the competition, businesses need to be responsive to ever-increasing market demands. The ability to change and adapt at a rapid pace gives a business a competitive edge. If a business is unable evolve and innovate, they will become irrelevant and obsolete and won't survive in today's increasingly competitive landscape.

Metrics provide a way to create a baseline of where a business is today and that, in turn, helps create a roadmap of process improvements to better streamline business operations. By tracking the metrics outlined above, a business can assess performance and identify bottlenecks that can be detrimental to the bottom line. To read more about companies that have evaluated and addressed key bottlenecks in their software release process, download our Case Study Bundle. With a focus on the proper metrics, your business can eliminate inefficiencies to remain competitive and ensure long-term survival.

www.liquibase.com | info@liquibase.com