Hewlett Packard Enterprise

Business white paper

# Measuring DevOps success

How do you know DevOps is working? Watch these KPIs.

# Table of contents

## Manage today's DevOps with metrics

In today's era of end-to-end DevOps, you need to measure these key performance indicators (KPIs) to demonstrate success and drive further transformation.

### DevOps without tradeoffs

DevOps originally grew out of the Agile movement as a way for development teams to speed up software delivery. As enterprises have embraced DevOps, it now includes every part of the software delivery process, from planning to securing, monitoring, and updating an application that's already in production.

Today, DevOps is widely accepted as the way that businesses of all kinds will build, test, and deliver software in the future. In this mobile, competitive, always-connected world, DevOps empowers enterprises to keep up with the rapid pace of delivery that users demand by breaking down the silos between development and operations.

These same conditions have also made quality more important than ever, so businesses can't let quality slide just to bring software to market quickly. There's no faster way to tank your brand's reputation than to release a poorly performing application that racks up negative reviews in an app store. With DevOps, optimizing for speed gives you quality and lowers your risk. Small batches, shipped faster, inherently reduce problems with quality and security.

By contrast, in traditional software development, increasing velocity often results in decreased quality and increased vulnerability. Similarly, improvements in velocity, quality, and security are traditionally understood to increase expense. But DevOps initiatives can actually drive improvements in all four dimensions: velocity, quality, productivity, and security. The key is adopting a metrics framework that indicates where you're succeeding and where you still need to improve.

64%

Number of IT decision makers in a recent survey who said that software was a key enabler for the business.[1]

[1] Forrester Consulting, "Application Delivery Speed Drives Success: How Mastering DevOps Enables Speed With Quality and Low Cost," September 2015.

## Metrics are essential

Most organizations implement DevOps because of a specific business need, such as improving time to market, reducing defects, or better aligning IT initiatives with business strategy. But because DevOps has no formal framework, there are few standard ways to measure DevOps success; and measurement is critical. Because many organizations implement DevOps gradually—first in a project, then slowly expanding throughout the organization—those first few projects must demonstrate success so they can generate executive support for DevOps. As our customers have told us, "What gets measured gets done."

Demonstrating metrics from a successful pilot can also help overcome internal resistance to DevOps adoption. A recent Gartner survey of IT and business leaders revealed that "people issues" were the biggest challenge organizations faced in adopting DevOps, with 43 percent of respondents citing resistance to change as the biggest inhibitor.[2] Because DevOps requires cultural transformation along with technology and process changes, executive support is critical.

In addition, DevOps is widely understood to be a journey that requires continual refinement. But if you're not measuring results, you can't understand how to evolve DevOps throughout your organization.

Metrics are essential to your DevOps journey. But how do you measure DevOps success? What marks a high-performing organization? Which KPIs can tell you what's working and what's not—and lead you to the insight that will explain why? What do you measure, how do you measure it, and what do the numbers really say?

This white paper outlines eleven KPIs that will create the foundation of DevOps metrics, a discipline that will continue to evolve as DevOps methodology becomes more ingrained in enterprises of all types.

50%

Percent of IT and business leaders who said that "people issues" were the biggest challenge to adopting DevOps.[3]

[2] Gartner, "Survey Analysis: DevOps Adoption Survey Results," September 2015.

[3] Gartner, "DevOps Adoption Survey Results."

## The four dimensions of DevOps metrics

To determine the best KPIs for measuring DevOps, **Hewlett Packard Enterprise (HPE) Software Services** first examined the top software-delivery challenges that customers report:

- Slow time to market
- Poor user experience
- High cost
- Poor predictability
- Vulnerabilities and risk

Any one of those challenges by itself would impede an enterprise's ability to meet user demands. But nearly every enterprise faces at least three or four of these challenges, and many enterprises face all five. It's no wonder that many software teams don't know where or how to start fixing their problems.

HPE Software Services determined that the root causes of these issues boil down to four areas:

- Velocity
- Quality
- Productivity
- Security

DevOps happens to address all four without tradeoffs. Proving success in each of those four areas will almost certainly improve time to market, the user experience, cost, predictability, and security.

HPE Software Services has taken the lead in defining the metrics that today's enterprises should be tracking. Some of our largest DevOps customers have already embraced and implemented this framework.

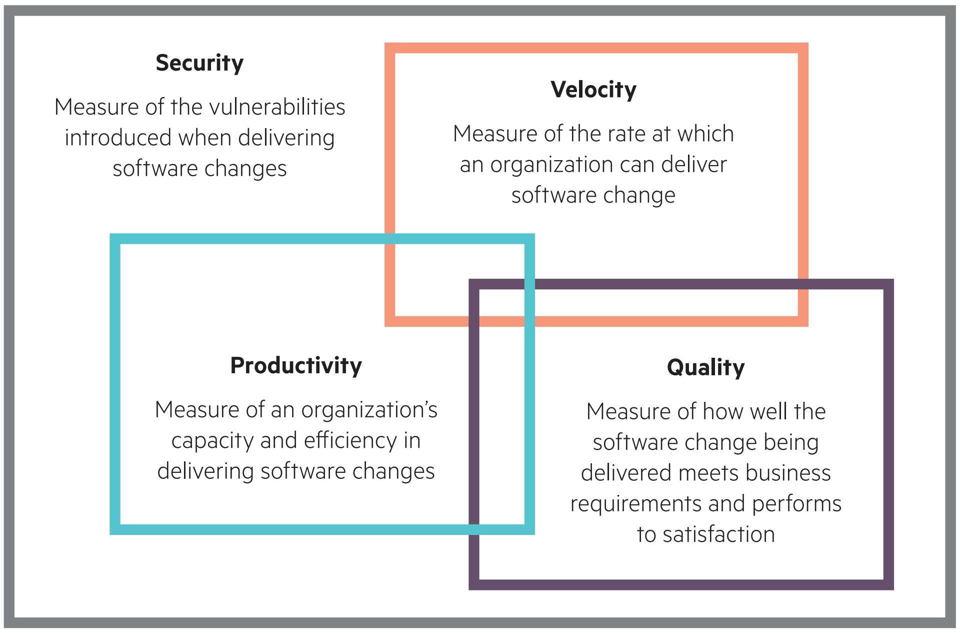
Security
Measure of the vulnerabilities introduced when delivering software changes
Velocity
Measure of the rate at which an organization can deliver software change
Productivity
Measure of an organization's capacity and efficiency in delivering software changes
Quality
Measure of how well the software change being delivered meets business requirements and performs to satisfaction

**Figure 1.** The four dimensions of DevOps metrics

Velocity, quality, productivity, and security are interrelated. Improvement in one area results in improvement in the other three, without tradeoffs.

**Why it's important to measure velocity**

The most important feature of DevOps is speed. DevOps was created as a way for developers to get features to market quickly in response to user demands for ever-faster, ever-better content. And while speed is relative—you'll probably always need to get faster and faster—most developer teams don't know what they're aiming for.

By tracking speed, you'll know whether you're getting faster or slower or are stuck in the same pattern. You'll also help the business more clearly understand IT's responsiveness.

Speed not only provides competitive advantage for the company, but also improves how business works with IT.

**Why it's important to measure quality**

According to a Forrester Consulting survey[4] of North American and European IT decision makers, 78 percent of respondents cited improved quality as a chief demand of business groups.

With DevOps, you don't have to sacrifice quality for speed. To increase speed, you break down software delivery into smaller units, so that you're delivering micro services with fewer or no dependencies. Smaller units mean faster delivery with better quality and reduced risk.

In addition, DevOps builds quality into the entire software delivery chain through a culture of excellence and the tools to support it. Culturally, DevOps emphasizes shared responsibility, expecting quality from developers, testers, operations and security team members, and everyone in between. Technologically, DevOps entails automating as much as possible to speed up processes and reduce human error. And a shift-left approach (such as shift-left security, testing, and monitoring) helps improve quality.

Quality isn't something you can guesstimate. It has to be defined by specific metrics and tracked accordingly.

**Why it's important to measure productivity**

The entire purpose of DevOps is to help IT create more value for the business. But this won't happen if your efforts aren't properly aligned with the business, no matter how fast you can churn out high-quality software.

When DevOps is done right, organizations will see increased velocity and increased quality, **without** increased costs. In fact, by shortening cycles, you will actually see increased business value. Tracking productivity metrics provides IT with a feedback loop. You can focus on building features that your users consume, thereby reducing waste and increasing business value.

**Why it's important to measure security**

Speed is supremely important in DevOps, but in today's world, so is security. Teams must ensure they don't introduce vulnerabilities when making changes.

Releasing new features that introduce new vulnerabilities or expose existing ones is a sure way to dissatisfy customers. Not only that, but vulnerabilities bring the release train to a grinding halt as teams scramble to fix them.

Like quality, security lends itself to a shift-left approach. It's far less costly (both in time and in real cost) to catch vulnerabilities early. When you perform shift-left testing, security testing is done early in the development cycle. Measuring security vulnerabilities early ensures that builds are stable before they pass to the next stage in the release pipeline.

In addition, measuring security can help overcome resistance to DevOps adoption. As vulnerabilities decrease, you're able to demonstrate that increasing velocity does not degrade a product's stability.

[4] Forrester, "Application Delivery Speed Drives Success." October 2015.

## Eleven DevOps KPIs

How do you measure velocity, quality, productivity, and security? These 11 KPIs create the basis for evaluating DevOps success.

**1. Frequency of deployment**
**What it means:** How often you release a new version of a specific product or service as measured by number of deploys per month, week, day, or hour.

**Why it matters:** Frequent deployments suggest continuous improvements to apps, which keep them competitive with fresh features. Also, frequent updates suggest that IT is responsive to business requests for new features. The business will have a new appreciation of IT if it knows that it can see new features within a month, as compared to six months. (Too frequent deployments, however, may indicate that you're rushing defective software into production.)

This metric also points to other areas of software development that you're getting right:

- Better architecture design at an earlier phase of DevOps: Modularity enables frequent deployment.
- Better testing coverage: If you're consuming more builds, operations will feel confident in deploying to production more frequently.

**2. Speed of deployment**
**What it means:** How long a single deployment takes for a specific product or service, expressed as number of hours from deployment approval to running in the next environment.

**Why it matters:** Traditionally, enterprise systems have been tightly coupled; every time you deployed something, you might end up redeploying the entire application. A decade ago, that was acceptable—you could schedule five-hour downtime windows for middle-of-the-night deployments without negative consequences to the business. But today's business is global—nighttime in the U.S. is daytime in Asia—and most enterprises have a customer base that's always awake somewhere. So speed of deployment becomes critical.

Today, systems need to be more loosely coupled. Through loose coupling, specific services can be changed without impacting the entire system. Using micro services and clearly defined APIs that enable encapsulation and easy access, you can greatly improve your speed of deployment.

In addition, increased speed of deployment reflects increased consistency throughout the DevOps pipeline, with automation removing manual, error-prone work from the lifecycle.

**3. Speed of build verification (QA)**
**What it means:** The total time for a build to go through the QA cycle.

**Why it matters:** QA can often become a bottleneck. Measuring how long it takes a build to go through QA provides key insight into how fast and how often you are able to release a product or service. When this metric is high, you know that you can increase frequency without QA becoming a bottleneck. If it's low, that's a sign you may need to implement techniques like test automation across the release pipeline.

**4. Frequency of build verification (QA)**
**What it means:** The number of builds a QA team can consume in a given period.

**Why it matters:** Increasing the number of builds that QA can consume positively impacts the overall release schedule. Your development speed might be high, but without continuous integration, you must handle builds manually, which slows down your release pipeline. Similarly, without continuous testing, time is wasted during the testing process. The result is that the QA team can't consume the maximum number of builds per cycle, and test coverage—as well as the confidence to move to the next level—suffers.

Continuous integration and continuous testing can improve your build verification frequency—in some cases, roughly doubling builds per month. When this metric is higher, it indicates better test coverage, a higher confidence level going into production, and faster time to production.

**5. Deployment success rate**
**What it means:** Percent of deployments that are considered successful after validation.

**Why it matters:** A high success rate when deploying to non-production servers implies high quality in each previous step. A low success rate suggests breakdowns in the coding quality. In many enterprises, you can have a high success rate in SIT and QA but still fail in staging. This metric can help you target and weed out problems at the development stage. Over time, the success rate should increase.

In top-performing organizations, this metric should be high as a result of eliminating problems in staging—or, better yet, avoided altogether. But many organizations are challenged in this area and must roll back releases. A low success rate after a successful staging phase can indicate several issues:

- Lack of consistency across environments
- Lack of good test data management
- Low testing quality and coverage

**6. Incident/defect volumes**
**What it means:** The number of incidents and defects reported in a specific release of a product or service.

**Why it matters:** Incidents and defects represent lost time, money, and opportunity. A high number here indicates problems in a number of areas:

- Quality of the requirements
- Quality of development
- Quality of testing, including test data management capability
- Level of collaboration between development, QA, and operations (for example, misunderstandings about requirements)

**7. Requirements coverage ratio**
**What it means:** The percentage of traceability between requirements and tests.

**Why it matters:** Inconsistency between requirements and test cases generates issues and waste during the software development lifecycle. When requirements and test cases are not linked, changes in requirements may lead to failed test cases or no test cases at all. Test cases won't show how many requirements have been covered. And when test cases don't cover all requirements, there is a direct, negative impact on software quality, and ultimately, the user experience.

When requirements and test cases are in sync, changes to requirements result in corresponding visibility in test case coverage. An update to any test case links back to requirements. The linkage directly and positively impacts the quality of the software. A higher ratio indicates improved software quality; less conflict between business analysts, developers, and the QA team; and higher efficiency.

**8. Feature usage**
**What it means:** The number or percentage of unused features in a specific product or service once it is in production.

**Why it matters:** DevOps is a feedback loop that includes the business, development, testing, operations, and users. Low usage means that IT is busy developing features that either aren't wanted (poor feedback from the business to development), aren't needed (poor feedback from users to the business), or are poorly created (a breakdown in feedback on the IT side). The cost of unused features can be tremendous, so it's important to determine the cause. A high number of features not used in production reflects negatively on the effectiveness of requirements development and management.

**9. Mean time to restore service (MTTRS)**
**What it means:** Time to restore a service or a function when the disruption is due to a defect that requires a fix to be developed. In other words, once you find an error, how long does it take to develop a fix and roll it out to production?

**Why it matters:** If you release an application with all of the required features and functions but find that it's unreliable or takes a long time to recover after a failure, you've destroyed all the value you created by building the right features and functions in the first place. You can't avoid every failure, but you can improve the resiliency of your applications and the way your team responds. For every minute that a service is down, you're losing revenue, customers, and brand equity.

In addition, MTTRS shines a light on related issues within your organization:

- The health of the feedback loop between the business, development, QA, operations, and end users
- The level of enterprise agility and how quickly the enterprise can react to changes from a people, process, and technology perspective
- Overall productivity and effectiveness

**10. Security test pass rate**
**What it means:** The ratio of failed-versus-pass static security source code scans after a successful build in a period of time.

**Why it matters:** Catching security vulnerabilities in the build stage is a critical part of guaranteeing that you always provide green (good) builds before triggering other parts of the DevOps release pipeline.

**11. Code scanning detection rate**
**What it means:** The number of security scans that come back with a problem in a given timeframe or given process phase, as well as the number of problems.

**Why it matters:** This rate should decrease with time or with movement from one stage to the next in the pipeline. Seeing the security scan success rate improve increases confidence that the speed at which new features are developed and released is contributing to the stability of the product rather than hurting security.

## Getting the numbers right

DevOps has the potential to transform the software delivery chain into a lean, fast, highly effective machine that delivers what users want. This transformation couldn't be more important. According to a recent Forrester Consulting survey[5], 64 percent of respondents claimed that software is the key enabler for their business and that their success depends on high-quality applications that enable modern business models.

HPE Software solutions powered by Big Data automatically gather metrics from diverse sources across your IT systems to build these 11 KPIs and more. They then consolidate the information, link it to a service hierarchy, and present insights in near real time through business-friendly scorecards and dashboards that help you convey the value of your technology—and, most important, the success of DevOps in your organization.

HPE Software solutions automate the collection and analysis of data from multiple sources—including non-HPE tools—and provide a single view of IT metrics in ways that any user, from service owners to business analysts to CIOs, can understand at a glance. They turn the often-opaque process of DevOps adoption into a transparent, easy-to-understand transformation and map out continued improvements over time. They arm IT with the data it needs to prove the worth of DevOps in clear, objective measurements that translate into time and money—exactly what executives want to see before getting onboard.

The innovation in HPE Software solutions provides a single view into DevOps metrics, helping you gain insights into your DevOps journey so you can effectively manage your DevOps transformation.

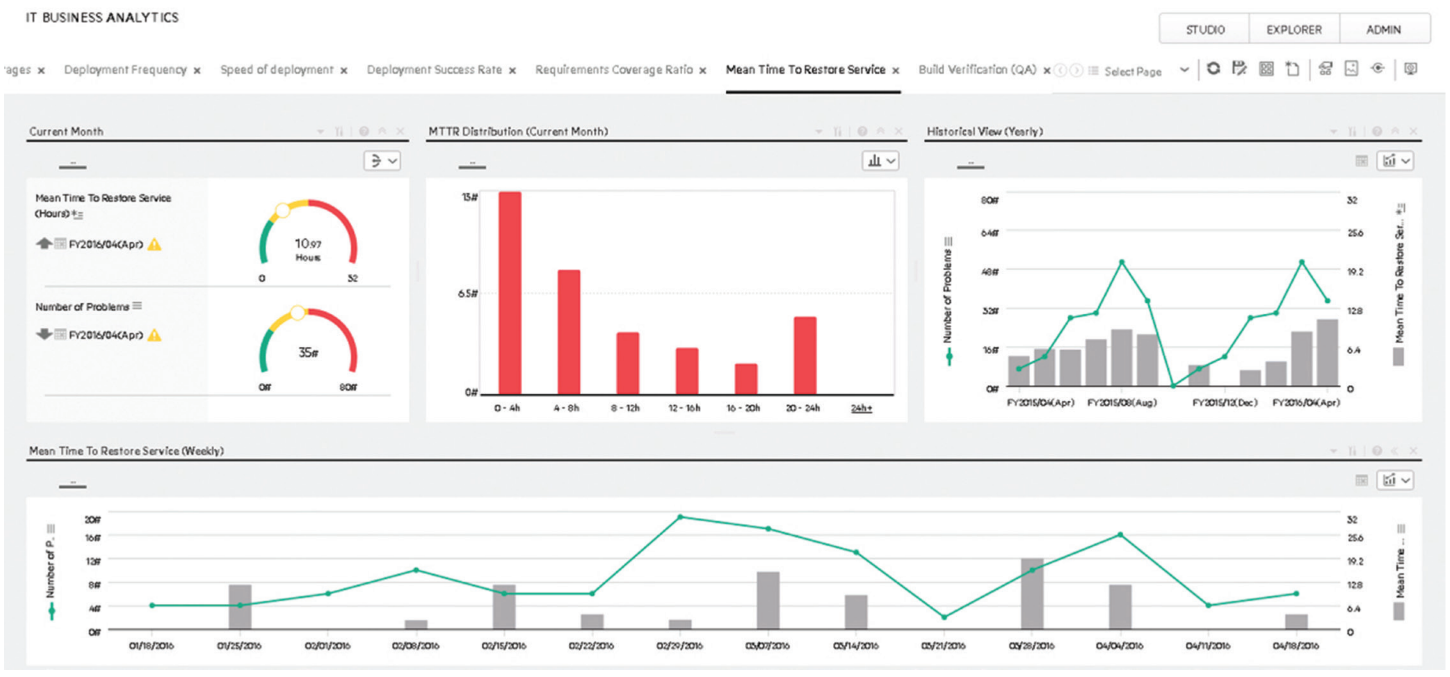

**Figure 2.** An example of an HPE Software solution dashboard

With HPE Software solutions, you have a clear view into DevOps metrics, such as MTTRS.

[5] Forrester, "Application Delivery Speed Drives Success." October 2015.

**Future directions for DevOps KPIs**
Once you start using KPIs to measure DevOps success, there are many ways to expand your metrics dashboard to make it relevant to other levels of your organization. To extend DevOps metrics up into the business domain, your DevOps KPI dashboard can roll up to a CIO dashboard that provides both a high-level view of the IT organization as well as progress on major initiatives. One current KPI that is of particular interest to the business is feature usage, but you could also track cycle time, indicating how long it takes for a requested feature to become available in production. Of course, pure financial KPIs are always relevant to the business—they form another avenue for expanding your dashboard.

In addition, you can extend your DevOps dashboard down a level to cover the day-to-day work of your operations practitioners.

"IT organizations must recognize that DevOps is transformative, and invest time and money in the people and process aspects of their efforts accordingly. IT leaders must not overemphasize technology at the expense of the other areas."

– Gartner, "Survey Analysis: DevOps Adoption Survey Results"

## Using metrics to continue the DevOps journey

DevOps is a journey, not an overnight implementation. Although many initiatives start small, DevOps requires a commitment to a new sort of culture. That's why organizations looking to implement DevOps today will likely need to make the business case to the organization's leaders. Other organizations that have already experimented with DevOps might be ready to start collecting data and using it to optimize their approach. In each case, highly visible, useful metrics will become an essential part of DevOps success.

Learn more at
**hpe.com/software/agilityservices**
**hpe.com/software/devops**

Hewlett Packard
Enterprise

4AA6-3036ENW, August 2016, Rev. 2