# CONTINUOUS DELIVERY METRICS

The DevOps approach to the software development lifecycle (SDLC) was introduced with the goal of optimizing the velocity, quality, and performance of the software development and delivery process.

The approach relies on best practices regarding:

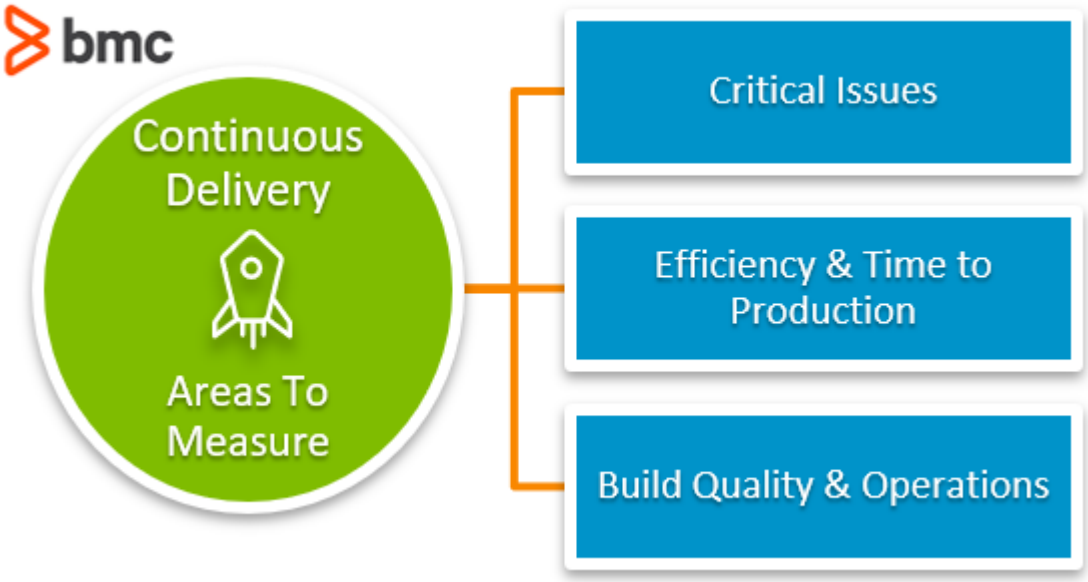
- Software engineering
- Organizational culture
- Business operations
- End-user interactions

In DevOps organizations, these framework guidelines are adopted by devs, IT Ops, QA, InfoSec, and other teams collectively working toward the common goal of delivering high quality software with every release as part of the Continuous Delivery (CD) practice.

So, how do DevOps teams evaluate their performance in terms of achieving these goals?

In this article, we'll look at some common metrics used to evaluate continuous delivery performance in DevOps.

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




## Critical issues

The first step to managing an efficient continuous delivery cycle is to eliminate the most critical and impactful risks. These risks include vulnerabilities in the network and technologies underlying the IT environment that can potentially compromise the software and data assets before they are even released to the market.

Example KPIs include:

- Number of critical vulnerabilities across all apps and services
- Network intrusion attempts
- Unauthorized traffic usage or data transfer attempts
- Number of failed security tests
- Percentage of security audits passed

Integrating security within the pipeline for Continuous Integration (CI), a subset of CD, is often challenging and not always welcome by devs pursuing faster CD cycles. As such, security must be built from the ground up at earlier stages of the SDLC.

## Efficiency & time to production

The primary target of a dev team is to ensure that the most functional software build is available for release within short CD cycles. The goal of ITOps is to ensure that the infrastructure resources are always available and consumed optimally for maximum cost efficiency of the process.

Together, Dev and Ops can look at the following KPIs to understand the performance of their continuous delivery cycles:

- **Number of code branches.** This refers to the number of feature components in progress. Unnecessary components add to the development burden and security risks.
- **Lead time to production/cycle time.** The time it takes to implement, test, and deliver a new build. This KPI depends on several variables, such as risk, technical debt, monitoring metrics, and other dependencies that may affect the lead time across teams, versions, and the state of

SDLC maturity.

- **Time To Value (TTV).** TTV evaluates the release process instead of the entire CD cycle. It refers to the time between writing code and releasing it. In other words, it looks at how quickly the end-user gains value from a new software build.
- **Team retention rate.** Used as a soft metric to evaluate the organizational culture, the team retention rate can show whether employees enjoy working at the targeted pace of the CD cycle. The average time that an employee spends on the DevOps teams may have an answer.

# Build quality & operations

Infrastructure performance is one of the top priorities of DevOps teams. When the software build is deployed to the test or production environment, it leaves a vast trail of log metrics at every node of the infrastructure. This information helps to:

- Understand the health of the underlying infrastructure
- Optimize application
- Identify potential security loopholes

Some metrics that can help achieve these goals include:

## Infrastructure utilization

Are some nodes oversubscribed? Which nodes show critical errors over a given time duration? Analyzing infrastructure utilization rates across all nodes, servers, virtual machines, and other hardware and virtualized components across the network can help you understand the true performance with every software build version.

## Production downtime & MTTR

Service level agreement (SLA) metrics pertaining to the availability, performance, and response of the CD cycle should be specified and applied for an application service. Continuous feedback and improvements can be used to improve the maturity of the continuous delivery cycle.

## Number of errors/error rates

This can be used to identify bugs and production issues in the new builds. These errors require a thorough evaluation of the cause and can depend on a variety of factors.

## Testing statistics

The number of tests, CD pipeline vs completeness of tests, and test coverage can help you understand the quality of software build as well as the efficiency of the testing process.

## Tickets handling time

Effective ticket resolution is not just about solving a specific problem facing an end-user but is primarily focused on understanding the problem's root cause. A large number of repetitive tickets, for instance, provide insights into specific recurring problems associated with the technology, app performance or control the workflow of the SDLC process.

## Continuous delivery requires measurement

DevOps organizations are adopting continuous delivery practices as an engineering principle. Delivering high quality software builds within a continuous process requires organizations to:

- Establish baseline metrics
- Adopt DevOps and Agile methodologies
- Reiterate by taking advantage of data driven insights from data

Finally, the improvements can be observed through tangible changes to the SDLC workflows and infrastructure & operations changes based on the continuous delivery metrics.

## Related reading

- BMC DevOps Blog
- How To Set Up a Continuous Integration & Delivery (CI/CD) Pipeline
- DevOps Metrics & KPIs
- Testing in DevOps: Concepts, Best Practices & More
- Deploying vs Releasing Software: What's The Difference?
- Software Project Management Phases & Best Practices