JANUARY 28, 2021

INSIGHTS SOFTWARE DEVELOPMENT

THE BUSINESS IMPACT & BENEFITS OF CI/CD

SEE TABLE OF CONTENTS OF RELATED ARTICLES.

CI/CD aims to enable software development teams to continuously deploy software updates into production to speed up release rates, reduce costs, and eliminate risks throughout the development process.

While cost-cutting might be the primary goal for many organizations, it's important to note that the business benefits of CI/CD extend well beyond the bottom line.

Manual processes are inherently error-prone and based on tribal knowledge. In other words, a lot can go wrong. Someone forgets to pass information to a new hire, skips an important step during the testing phase, or submits code containing one tiny mistake that slips through the cracks can turn into a big problem.

By adding automation into the mix, developers can spend more time on building high-quality apps that speak directly to customers' needs/expectations, deliver solutions to

Continuous integration and continuous delivery (CI/CD) help teams bring better software to market faster, work more effectively, and reduce risk. Organizations that understand the relationship between CI/CD and business value stand to reap major rewards/secure a competitive edge over those that fail to make it a priority.

Below, we'll look at what CI/CD means for the development team, QA, and other stakeholders. We'll also look into how the process supports client outcomes/strategic goals. You'll hear from 3Pillar experts who weigh in with their thoughts on why CI/CD is important on multiple levels.

What is CI/CD?

Continuous integration/continuous delivery (CI/CD) is a software development practice that combines development and operations teams and their day-to-day tasks. It applies automation to developing, testing, and delivering applications.

By unifying these processes under one strategy, CI/CD supports a smoother deployment process, brings more structure to the entire code and the development process, and enables more frequent updates with fewer disruptions.

While CI/CD is often discussed as a single strategy, CI and CD represent two distinct ends of the quality control spectrum. Here's a quick look at both concepts and how they fit together:

Continuous integration (CI) is a software development process where teams integrate code early and often into a central repository where they can run frequent tests and validate changes.

The goal is to speed up the release process by enabling teams to identify and fix

CI reduces the amount of time spent on bug fixes and regression testing and ensures that everyone has a deep understanding of what's happening inside the codebase and what features they're developing for end-users.

Continuous Delivery (CD) is the practice of getting all updates, fixes, features, and configuration changes either into production or into the hands of end-users as quickly (and safely) as possible.

The goal is to streamline the delivery/deployment process so that predictable tasks can be performed on-demand.

CD aims to ensure that the code is always in a deployable state, even with developers making continuous updates to the codebase by bringing integration and testing together.

The result is, releases become a routine affair that can be performed anytime new code is ready. This allows teams to streamline the development process, reduce risk, and implement user feedback more quickly. If any errors are detected during production, they can be addressed immediately by simply rolling out the next update.

CI/CD Pipeline. CI/CD pipelines are a series of steps that must be completed to deliver a new software release. The aim is to improve the software delivery process by introducing monitoring and automation to improve the development and delivery process.

The CI/CD pipeline typically breaks down into the following stages:

- Build
- Test
- Release
- Deploy
- Validate

While it's technically possible to execute each of these steps manually, it's important to note that the real value of CI/CD is realized through automation.

What are the Business Benefits of CI/CD?

So, what are the business benefits of continuous integration and continuous delivery? In

Bring Products to Market Faster

Organizations that have effectively implemented CI/CD can bring new products and features to market faster and immediately start generating revenue from the features they deploy rather than waiting for the entire app to be completed (and checked manually) before they can launch.

Instead, teams already know the code is in good shape because they've automated testing, and continuous delivery means code is automatically deployed if it meets predefined criteria. Back-to-back releases are easier and less time-consuming, and, if something isn't working, they can pull features with a single click.

Allows Developers to Deliver Products Consumers Want Now.

Over the last couple of years, customer-centricity has been a core focus for businesses.

CI/CD enables organizations to respond to consumer needs as they evolve.

Teams have the flexibility to update applications and build and deploy new ones in response to emerging trends, new markets, and evolving expectations. According to Rodolfo Carmona, "good CI/CD implementation starts early in the production process by making the core functionality available to end-users right away. That way, early feedback and usability issues can be addressed without the need for major time-consuming refactors to change the direction later on."

CI/CD plays a crucial role in shortening time to value.

According to 3Pillar Software Engineer, Paul Estrada, the strategy "fosters a culture of innovation, allowing developers to experiment with new technologies and try out new ideas. Teams can test different features with real users in parallel and use their findings to ensure

CI/CD supports customer outcomes from a technical standpoint.

3Pillar's Rong Wang says one of the biggest benefits of CI/CD is that it allows businesses to ensure customers receive a seamless experience and uninterrupted service. When a product goes to market, organizations typically monitor and record user activity and collect feedback from consumers. This allows them to make sure that software bugs and usability issues don't go undetected and cause lasting damage to the product's (or the company's) reputation.

organizations can continuously build on applications and enhance the experience without the risk of downtime or interruptions.

Boosts DevOps efficiency

Without CI/CD, developers and engineering teams deal with more pressure in their day-to-day—service interruptions, outages, and bad deploys can put their jobs at risk.

According to 3Pillar's Jorge Gaona, "devs need to write deployment scripts, documents, procedures, and so on, in order for the ops team to execute them. The process is inefficient and often risky."

CI/CD automation can eliminate manual tasks, prevent coding errors, detect problems before deployment, allowing teams to work faster without compromising quality. Because it takes less time for development teams to find and fix problems during the production process, CI/CD can dramatically accelerate release rates. What's more, organizations can support recurring releases if code is developed in an automated testing pipeline.

With CI/CD, teams can also implement a standardized delivery mechanism that automatically merge codes, run tests, and deploy changes to multiple production environments to ensure that code is always in a release-ready state.

CI/CD Improves App Quality

One of the biggest concerns organizations have about implementing CI/CD is that they're giving up quality in favor of speed.

As Paul Estrada puts it, "quality shouldn't be tied to timelines. If you sacrifice quality in the hopes of meeting a deadline, there's a good chance you'll find bugs in the software post-release."

Done right, CI/CD means organizations don't need to choose to prioritize quality over speed. Instead, it offers the best of both worlds by enabling tight collaboration between developers and operations teams, which, in turn, means problems are identified and fixed faster and early in the development lifecycle.

One of the most notable examples of CI/CD's business value is that it allows dev, ops,

environment.

Additionally, CI/CD helps organizations maintain quality standards as apps expand and evolve over time. Applications with large feature sets can get too big to feasibly run proper code review, meaning you may see a drop in quality over time.

CI/CD automation allows developers to deploy code more frequently and make incremental improvements to the code. Teams can automate regression testing and parallel tests, which improves test coverage and ensures that the code is bug-free and works across multiple environments.

What's more, it's much easier to roll back to the previous version without causing damage to other features and components in the case of failure.

Supports Cloud-Based App Development

The rise of CI/CD is, in part, due to the rise in cloud adoption. Technically, you could use traditional development techniques to build cloud-based applications, but it's not really practical.

If you're deploying cloud software using the same process as traditional, on-premise solutions, you're missing out on the benefits the cloud provides, like automation and tooling (APIs and CLIs).

CI/CD enables benefits like scalability, elasticity, improved performance characteristic of cloud-native applications.

Reduce Costs and Boost Profits

CI/CD is also good for the bottom line. It standardizes deployment processes across all projects, and, done right, it enables teams to systematically test every change made to the source code.

As a result, this process stands to dramatically reduce the likelihood that any bugs or errors slip through the cracks and cause problems down the line. Done right, this practice can lower development costs by eliminating many of the costs incurred while building and testing code changes.

And because CI/CD also makes it easier to deliver high-quality products to market faster and respond to feedback as it comes in, organizations stand to see an increase in profits. Customers stick around longer and will likely recommend your products to others in their network.

Gain Real-Time Visibility of the Development Process

CI/CD also brings real-time visibility into the development cycle. Reviewing test results helps everyone on the team identify the project status and immediately understand which code changes caused problems or improved on what came before.

Stakeholders can easily see where a project stands at any given moment—spot bottlenecks, inefficiencies, etc., and use those insights to optimize the process.

You can also see the history of deployments and success rates, which can inform the direction of future projects, help teams plan ahead, and keep everyone focused on the tasks that create the most value.

Set the Stage for Success

While this isn't a comprehensive strategy for implementing CI/CD, here's a basic overview of how you can make sure that your development team has what they need to succeed.

- **Make sure you nail CI before introducing CD.** Start with a strong foundation—consistent tooling, configurations, libraries, etc., should all be in place before CD enters the mix.
- **Separate services.** It's crucial that series are separate from one another/should be isolated and abstracted to ensure that changes made to one feature won't impact the others.
- **Automate Repeatable Processes.** While writing code is still a very "human" process, many aspects of the development lifecycle can be automated. CI/CD automation streamlines the software development cycle, allowing teams to work through the feedback cycle faster and consistently deliver high-quality products with fewer instances of code errors/bugs.
- **Invest in the proper tooling and infrastructure.** Make sure you invest in the underlying infrastructure and the correct tooling to support CI/CD. Need the right tools to expedite the process/eliminate repetitive manual tasks. While

allow them to innovate faster and respond to customers' needs.

Conclusion

The benefits of CI/CD impact all ends of the development lifecycle, the customer experience, and the big-picture business strategy.

It plays a critical role in software development and delivery and helps smaller teams move faster, respond to constant changes, and incorporate real time feedback—all of which contribute to cost savings, profitability, and a higher-quality end-product.

Software development is the heart of 3Pillar Global. But there is more. Our focus on the Product Mindset is what sets us apart from others. Contact us today to learn more.

Software Development Table of Contents

Ch. 1: Current State of Software Development

Ch. 2: Importance of Software Development

Ch. 3: 10 Leadership Traits for Modern Software Development Leaders

Ch. 4: Importance of Good Governance Processes in Software Development

Ch. 5: Insights for Recruiting and Retaining Great Software Talent

Ch. 6: Developing a Software Training Strategy That Rocks

Ch. 7: Best Practices for Creating a Software Culture Where Teams Can Thrive

Ch. 8: Building and Managing High-Performance Teams

Ch. 9: Critical Metrics for Measuring Software Development Team Performance

Ch. 10: Strategies for Improving the Performance of Software Development Teams

Ch. 11: Managing Distributed Software Teams in the Age of Covid-19

Ch. 12: Preparing for the Future with Application Modernization

Ch. 13: Creating an Agile Enterprise with Minimum Viable Architecture

Ch. 14: 10 Components of an Effective Change Management Strategy

Ch. 15: How to Choose the Right Cloud Platform

Ch. 18: The Business Impact & Benefits of CI/CD

Ch. 20: Everything You Need to Know to Create a Test Automation Strategy

Ch. 21: Developing an Enterprise Software Strategy

Ch. 22: Build or Buy? A Decision-Making Framework for Software Development Investments

Ch. 23: Strategies for Getting Approval for Software Development Projects

Ch. 24: 10 Software Development Challenges Faced by Modern Enterprises

Ch. 25: Software Development Best Practices

Ch. 26: The Most Common Reasons Software Projects Fail

Ch. 27: Measuring the Success of Your Software Development Projects

Ch. 28: The Importance of Integrating the User Experience into Software Development

Ch. 29: The Importance of Quality Attributes in Software Architecture

Ch. 30: Predictions About New Trends in Software Development

Back To Top