fastly

Why your continuous integration and delivery practices need a next-generation CDN

WHITE PAPER

Introduction

In today's highly competitive business environment, organizations are under constant pressure to innovate. Whether it's pushing out new features or supporting rapid content releases, there is a growing demand to be able to iterate more rapidly.

To meet this demand, most engineering organizations have already adopted processes like Agile and Waterfall. Yet, these processes fail to address all development bottlenecks. That's why more organizations are starting to embrace a new process — continuous integration and delivery (CI/CD).

In this brief we will examine the key elements of CI/CD. We will consider the benefits and challenges of this new approach, from both a technical and cultural perspective. We will also explore how your content delivery network (CDN) can be a critical part of your CI/CD process, and why it's important to look for a next-generation CDN that can complement your software development efforts.

Continuous integration and delivery and the role of your CDN

Older development methods rely on daily, weekly, or monthly integration following completion benchmarks. While these methods seem straightforward for scheduling purposes, they can result in unnecessary complexity and unpredictability when it comes time to push new code.

By contrast, continuous integration is seen by many as a natural evolution of the integration process. CI merges source control and frequent commit practices with automated builds and testing. This allows for high-velocity development while maintaining control and visibility by setting clear procedures for source access and regular updates. Continuous delivery is the automated publication of that code to the production environment.

To be successful, CI/CD should automate workflows across your entire development stack. Ideally all third-party vendors should offer capabilities that create a seamless extension of your development environment. This is especially true of your CDN provider. Given where it sits between your platform and your end users, it's critical to choose a CDN that can complement rather than impair your CI/CD efforts.

The elements of continuous integration

At its heart, continuous integration consists of three elements: source code management, commit cycle discipline, and build automation. With these simple practices, multi-origin changes can easily be incorporated into stable builds, with fewer bugs and minimized developer time. CI eliminates the risk of an end-cycle integration tangle, protecting your deployment timetable.

Source code management – everything but the build, in a single, easy to access system

The source repository forms the core of continuous integration. The development process involves a lot of moving parts, and complexity scales with team size and project duration. By concentrating on a single repository, you avoid potential code divergence and the associated frustration.

Continuous integration relies on a transparent and easily accessible repository, which in turn requires a properly configured source code management (SCM) tool. SCM systems provide a stable and comprehensive platform for project development. They expedite version control, manage file access, and naturally store all vital project files, which includes the operative code and all the associated scripts, libraries, databases, and integrated development environment (IDE) configurations. With an SCM, the development team has a common resource for all necessary files. Some popular CI and SCM tools include:

- **Jenkins (Java CI tool)** – Jenkins is a server-based, cross-platform CI tool with broad plug-in support for a variety of builds and databases. Jenkins offers a wide range of build options, and is available for use under MIT's free software license.
- **CircleCI (commercial CI tool)** – CircleCI is a relative newcomer to the CI market, and has attracted several high-profile clients with its reputation for speed and ease of use.
- **Travis CI (Git-focused CI tool)** – Primarily intended for use with Git projects, Travis CI is a hosted service for CI implementation. Travis is notable for its advanced testing capabilities and diverse language support.
- **Git (open source SCM)** – A popular option for development, Git is a free and open source SCM with a variety of configurations. Git excels in distribution, code review, and technical performance, and has excellent tools for managing multiple workflows. For first time users, the complexity of Git's options can be a barrier, but there is a wealth of documentation available.

Whatever you choose, it's important to automate the flow of code from your SCM tool to your CDN and vice versa. Without a tight integration, companies risk slower and less reliable development cycles due to manual processes and human error.

The commit cycle – quality from consistency

Even the best SCM system is worthless if it isn't being used. Continuous integration needs the active participation of the development team. If developers are working in isolation, or not frequently committing their (build-tested) code to the repository, the result is a messy integration process and unnecessary errors.

A disciplined commit process surfaces errors and conflicts immediately, and minimizes redundant or non-viable work. Frequent commits also encourage productive coding behavior, helping developers divide up their work into meaningful elements. By pursuing concrete objectives, teams gain a sense of accomplishment, and deliver more cohesive code. Modern CDNs makes it easy for developers to commit their code multiple times a day, with real-time visibility and instant invalidation.

Automation and testing – better builds by design

The true benefits of continuous integration rely on smart automation and testing. As your developers access the repository and introduce new code into the source, they must navigate dozens or even hundreds of procedural elements: loading database schemas, applying naming formats, changing file locations, compiling files, checking dependencies, and more.

These actions are essential for a cohesive system, but are needlessly repetitive and often lead to human error. Nearly all of these build procedures can use automated environments and basic script tools. With automation, you can cut any busywork from the CI process and enjoy a more streamlined development cycle.

The network edge presents huge opportunities for automated testing. Look for a CDN that can push network logic to the edge, so you can automatically pass validated configuration settings from development environment to production without human intervention. Due to the frequent commits required by CI, self-testing code can catch preliminary errors before they become integrated. By adopting these advanced screening techniques, the overall build pattern becomes more agile and robust.

A step beyond – continuous delivery for push-button release

Leading, innovative companies take the CI model a step further, following a practice known as continuous delivery (CD). CD emphasizes not just integrated code, but release-ready code, at every iteration. Use of this process attempts to erase the line between development and production by focusing on the ability to constantly deploy.

The benefits of CD center around the constant availability of a finished product. The production-like environment promotes calculated risk management, as any errors can be quickly identified and isolated thanks to structured releases.

Most importantly, CD incorporates the end user directly into the development cycle. While rigorous testing in development will likely catch most errors, active and passive user feedback is crucial for identifying production-level issues. Any bugs can be resolved immediately so end-user experience doesn't suffer.

Real-time analysis and configuration options are essential for achieving the benefits of a CD platform. Successful use relies on absolute version control, which is impossible without instant configuration changes and effective rollback capacity. And the value of user information stems from the ability to monitor and analyze data in real time. Deep control and configuration agility give your company a platform for quick response.

While this set of practices has yet to gain widespread adoption, CD highlights the potential of automation and immediate feedback for continuous, incremental improvement.

The benefits of continuous integration and delivery

To this point we've addressed CI/CD, but equally important is why it matters. By adopting this methodology, a company can gain a wide array of benefits, some immediately obvious, some revealed over time. Here are a few of the principal reasons why CI/CD is worth the effort.

Single platform management – As touched on above, by centralizing project resources within a single source, companies can more efficiently manage their IP.

Multi-platform operations inevitably result in unnecessary redundancy and wasted work. With a single platform, errors can quickly be revealed and fixed, updates can be deployed system-wide, and developers can easily monitor project metrics.

End-user empowerment – Beyond the obvious advantage of fewer error messages, end-users benefit from a more interactive production. For software with update schedules, continuous integration means fewer delays, and more efficient distribution. And with user feedback in hand, developers are able to regularly incorporate change requests to meet market demands.

Optimized development cycles - Code that is quantifiable is optimizable. By breaking projects into segments based on an update schedule, CI/CD facilitates better organized development and production environments. Progress can be measured, errors can be addressed immediately, and workflow becomes more stable.

Agile in action – Integration barriers disrupt delivery, throwing a wrench into your company's business. CI/CD allows product to keep pace with engineering innovation. Restrictive release windows can suffocate responsive business practices, whereas CI/CD keeps your development cycles in sync. Additionally, SCM easily allows for widespread collaboration, and ensures all parties are on the same page.

Risk mitigation – Newcomers to CI/CD are often daunted by the potential error risk they see in frequent code integration. However, CI/CD actually helps remove the errors predominant in traditional integration methods. By consistently and regularly integrating code within a common source, any errors that crop up can be immediately isolated and fixed. CI's self-testing, automation, and frequent evaluations keep code clean and coherent, while CD ensures that your end users always have the most up-to-date software.

The challenges of continuous integration and delivery

Given that continuous integration and delivery represents a shift in development methodology, there are some challenges inherent to its execution. However, these issues are largely tied to workplace culture and less related to technology. The four biggest hurdles to CI/CD implementation are:

Initial developer hesitance – After performing a task a hundred times, there is always some resistance to change, no matter how beneficial. Developers accustomed to older integration strategies may be reluctant to abandon previous approaches. CI/CD practices are easy to adopt once the source backbone is in play, and the time-saving benefits quickly become apparent and very persuasive.

Commit discipline – Once CI/CD practices have been implemented, their effectiveness relies on consistent application. If developers abandon the frequent commit procedure, or skip testing, projects can revert to confusion. By heavily incorporating automation, the process can be made much easier for participants, and CI/CD quickly becomes a valuable habit.

Black box vendors – While many of the most popular options for continuous integration systems are open source, some traditional vendors are less transparent. Be sure to select vendors and software that is transparent and accessible, or you might find your newly vital system is outside your control.

The ability to instantly modify the configuration of any third-party vendor in the CI/CD workflow is also critical, and the use of such vendors extend the resources available to your developers.

End-user experience – When CI/CD systems are poorly designed or inefficient, the end-user suffers. Previous models were hampered by system traffic and delays. By extending CI/CD to a modern CDN with real-time configuration changes, you can ensure your users have clear and consistent access that won't be disrupted.

One obsolete challenge – the CDN tradeoff

There is an outdated belief that a tradeoff must exist between user performance and real-time control, but this is no longer true. With traditional, black-box CDNs, CI/CD suffers from configuration delays, unpublished release queues, and non-purgeable content. This issue has been resolved in next-generation CDNs, which offer customers granular control, instant invalidation, and real-time visibility of their content at the edge.

Choosing a CDN for continuous integration and delivery

When considering CDNs for CI/CD, it's vital to consider details beyond dollars per gb/s. The CI/CD development cycle depends on a CDN with control and configurability in order to properly mesh with your workflow. A CDN shouldn't be a barrier to DevOps teams, but rather so responsive, configurable, and automated that it naturally becomes part of the development environment. Here are four factors to look for in a CI/CD-ready CDN:

- **'API first' approach** – The biggest gains from CI/CD implementation can come from the ability to execute changes without human intervention. Your CDN should offer a robust and flexible platform for autonomous customization, without sacrificing performance.
- **Granular control and real-time visibility** – Next-generation CDNs are a natural extension of your development environment. These systems rely on responsive monitoring and in-depth control.
- **Seamless version control** – To support CI/CD at the edge, you need to make adjustments instantly. Make sure your CDN supports configuration changes and rollbacks in mere seconds, or you could be caught waiting indefinitely.
- **Proven platform with CI/CD customer use cases** – Look for examples of other innovative companies that are extending CI/CD to the edge with their CDN. Examples may include scripts written by CDN vendors or their customers that help to streamline and automate workflow.

Conclusion

Continuous integration and delivery gives your company the flexibility to meet the increasing pressure to innovate. It eliminates the old integration roadblock and helps you accelerate your development cycle. Instead of wasting time and money on conflicting code, you can focus on a direct path for your products. CI/CD provides a reliable and responsive methodology that cuts risk and results in better code.

To realize these benefits, you need a CDN that fits your workflow. When it comes to CI/CD, Fastly does what traditional CDNs can't. By providing a powerful platform that works with your current systems, and control over the details that matter, Fastly lets you embrace CI/CD without the wait.

More information

For more information on how Fastly can facilitate your continuous integration and delivery efforts, please contact sales@fastly.com.