Fundamentals and Benefits of CI/CD

I. CI/CD explained

CI/CD falls under DevOps (the joining of development and operations) and combines the practices of continuous integration and continuous delivery. CI/CD automates much or all of the manual human intervention traditionally needed to get new code from a commit into production such as build, test, and deploy, as well as infrastructure provisioning. With a CI/CD pipeline, developers can make changes to code that are then automatically tested and pushed out for delivery and deployment. Get CI/CD right and downtime is minimized and code releases happen faster.

1. What is continuous integration (CI)?

Continuous integration is the practice of integrating all your code changes into the main branch of a shared source code repository early and often, automatically testing each change when you commit or merge them, and automatically kicking off a build. With continuous integration, errors and security issues can be identified and fixed more easily, and much earlier in the software development lifecycle.

By merging changes frequently and triggering automatic testing and validation processes, you minimize the possibility of code conflict, even with multiple developers working on the same application. A secondary advantage is that you don't have to wait long for answers and can, if necessary, fix bugs and security issues while the topic is still fresh in your mind.

Common code validation processes start with a static code analysis that verifies the quality of the code. Once the code passes the static tests, automated CI routines package and compile the code for further automated testing. CI processes should have a version control system that tracks changes, so you know the version of the code used.

2. What is continuous delivery (CD)?

Continuous delivery is a software development practice that works in conjunction with continuous integration to automate the infrastructure provisioning and application release process.

Once code has been tested and built as part of the CI process, continuous delivery takes over during the final stages to ensure it can be deployed packaged with everything it needs to deploy to any environment at any time. Continuous delivery can cover everything from provisioning the infrastructure to deploying the application to the testing or production environment.

With continuous delivery, the software is built so that it can be deployed to production at any time. Then you can trigger the deployments manually or move to continuous deployment where deployments are automated as well.

II. CI/CD fundamentals

There are eight fundamental elements of CI/CD that help ensure maximum efficiency for your development lifecycle. They span development and deployment. Include these fundamentals in your pipeline to improve your DevOps workflow and software delivery:

1. A single source repository

Source code management (SCM) that houses all necessary files and scripts to create builds. The repository should contain everything needed for the build. This includes source code, database structure, libraries, properties files and version control. It should also contain test scripts and scripts to build applications.

2. Frequent check-ins to main branch

Integrating code in your trunk, mainline or master branch — i.e. trunk-based development — early and often. Avoid sub-branches and work with the main branch only. Use small segments of code and merge them into the branch as frequently as possible. Don't merge more than one change at a time.

3. Automated builds

Scripts should include everything you need to build from a single command. This includes web server files, database scripts and application software. The CI processes should automatically package and compile the code into a usable application.

4. Self-testing builds

Testing scripts should ensure that the failure of a test results in a failed build. Use static pre-build testing scripts to check code for integrity, quality and security compliance. Only allow code that passes static tests into the build.

5. Frequent iterations

Multiple commits to the repository results in fewer places for conflicts to hide. Make small, frequent iterations rather than major changes. By doing this, it's possible to roll changes back easily if there's a problem or conflict.

6. Stable testing environments

Code should be tested in a cloned version of the production environment. You can't test new code in the live production version. Create a cloned environment that's as close as possible to the real environment. Use rigorous testing scripts to detect and identify bugs that slipped through the initial pre-build testing process.

7. Maximum visibility

Every developer should be able to access the latest executables and see any changes made to the repository. Information in the repository should be visible to all. Use version control to manage handoffs, so developers know which is the latest version. Maximum visibility means everyone can monitor progress and identify potential concerns.

8. Predictable deployments anytime

Deployments are so routine and low-risk that the team's comfortable doing them anytime. CI/CD testing and verification processes should be rigorous and reliable. This gives the team confidence to deploy updates at any time. Frequent deployments incorporating limited changes also pose lower risks and can be easily rolled back.

III. Benefits of CI/CD

1. 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.

2. Automation reduces the cost

Anytime a human does not have to intervene in the software development process, time, and thus money, are saved. That's why automation is the underpinning to successful DevOps practices. CI/CD automates the handoffs, the source code management, the version control system, the deployment mechanisms, and, of course, so much of the testing.

Of all those, testing is arguably the most important. In our 2021 survey, testing was identified as the number one reason releases were delayed. Not only do delayed releases impact the business from a cost, branding, public relations, and even a reputation perspective, they are deadly to businesses relying on speedy time-to-market. Historically software testing was manual and incredibly time-consuming, which is why companies only released new code once or twice a year. In today's world, companies have to release all the time, and automated software testing is critical to making that possible.

3. Customer Satisfaction

The advantages of CI/CD do not only fall into the technical aspect but also in an organization scope. The first few moments of a new customer trying out your product is a make-or-break-it moment.

Don't waste first impressions as they are key to turning new customers into satisfied customers. Keep your customers happy with fast turnaround of new features and bug fixes. Utilizing a CI/CD approach also keeps your product up-to-date with the latest technology and allows you to gain new customers who will select you over the competition through word-of-mouth and positive reviews.

Your customers are the main users of your product. As such, what they have to say should be taken into high consideration. Whether the comments are positive or negative, customer feedback and involvement leads to usability improvements and overall customer satisfaction.

Your customers want to know they are being heard. Adding new features and changes into your CI/CD pipeline based on the way your customers use the product will help you retain current users and gain new ones.

4. Continuous feedback

A unified CI/CD process, operating as part of a DevOps platform, gives everyone on the team – including business stakeholders – a way to see what's happening, where it's happening, and what might be going wrong. This sounds like a simple thing, but in reality, a single window into software development is almost revolutionary.

In the past, there were simply so many tools in play that a project manager might have to look in a number of places, and ask a number of people, to get status updates. Developers and operations pros fared no better. Obviously that was a waste of time and resources, particularly when problems arose.

5. CI/CD supports customer outcomes from a technical standpoint.

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.

Frequent updates, new feature releases, rapid response to feedback, and quick bug fixes all play a major role in customer satisfaction and long-term loyalty. By using CI/CD, organizations can continuously build on applications and enhance the experience without the risk of downtime or interruptions.

6. 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.

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.

7. 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.

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, and QA to focus on what they do best. Developers can spend more time writing code that solves real problems rather than worrying about what's going on in the production 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.

8. 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.

9. 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.

Teams spend less time on testing and bug fixes, meaning organizations spend less money on tasks that don't provide any value to the business or its customers.

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.

10. 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.