CI/CD

Definition

CI and CD stand for continuous integration and continuous delivery/continuous deployment. In very simple terms, CI is a modern software development practice in which incremental code changes are made frequently and reliably. Automated build-and-test steps triggered by CI ensure that code changes being merged into the repository are reliable. The code is then delivered quickly and seamlessly as a part of the CD process. In the software world, the CI/CD pipeline refers to the automation that enables incremental code changes from developers' desktops to be delivered quickly and reliably to production.

What is the difference between CI and CD?

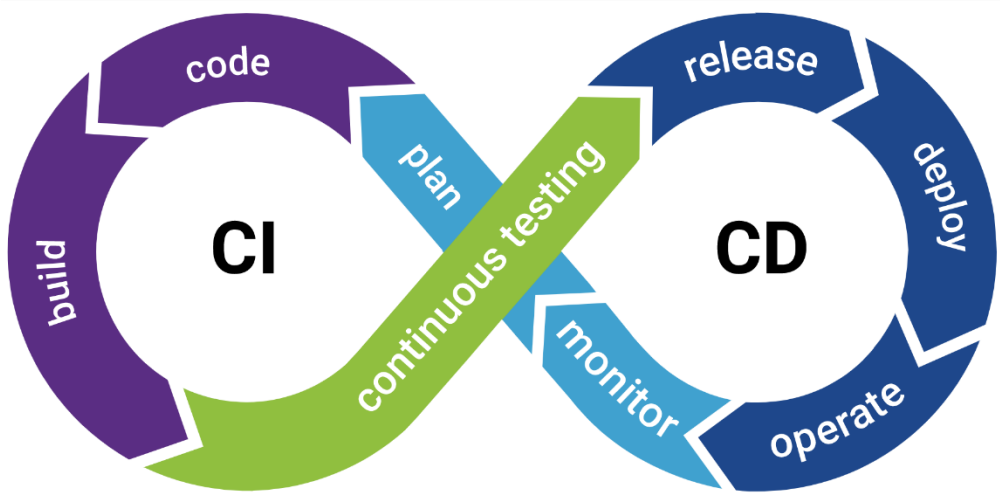
Continuous integration (CI) is practice that involves developers making small changes and checks to their code. Due to the scale of requirements and the number of steps involved, this process is automated to ensure that teams can build, test, and package their applications in a reliable and repeatable way. CI helps streamline code changes, thereby increasing time for developers to make changes and contribute to improved software.

Continuous delivery (CD) is the automated delivery of completed code to environments like testing and development. CD provides an automated and consistent way for code to be delivered to these environments.

Continuous deployment is the next step of continuous delivery. Every change that passes the automated tests is automatically placed in production, resulting in many production deployments.

Continuous deployment should be the goal of most companies that are not constrained by regulatory or other requirements.

In short, CI is a set of practices performed *as developers are writing* code, and CD is a set of practices performed *after* the code is completed.



What are the benefits of CI/CD?

1. Higher efficiency

Increased productivity is one of the leading advantages of a CI/CD pipeline. You should automate your process if you have a review process that includes deploying code to development, testing, and production environments and entering multiple commands across several domains. This creates the need for a CI/CD framework

2. Reduced risk of defects

Finding and resolving defects late in the development process is costly and time-consuming. This is particularly true when problems arise with features already released to production.

You can test and deploy code more frequently using a CI/CD pipeline, giving QA engineers the power to identify and fix errors as soon as they occur. This way, you're effectively mitigating risks in real-time

3. Faster product delivery

With a smooth CI/CD workflow, multiple daily releases can become a reality. Teams can automatically build, test, and deliver features with minimal manual intervention. Docker, Kubernetes, and Travis CI are some of the tools and frameworks that can be used to accomplish this.
CD enables your team to provide customers with frequent and timely updates.

When CI/CD is used, the entire team's efficiency increases, including the release of new features and fixes to problems. Businesses can address
market shifts, security challenges, consumer needs, and financial pressures faster.

If a new security feature is required, your team can
use CI/CD and automated testing to introduce the fix to production systems faster and w ith higher assurance. What used to take weeks or months to finish
can now be completed in days or even hours.

4. Log generation

Observability is pivotal for DevOps. If something isn't right, you need to figure out why. You'll need a way to track the system's performance over time to determine essential performance indicators. Observability is a technical tool that aids in this endeavor.

Logging information plays a vital role in observability. Logs provide a large volume of information to decipher what's happening beneath the UI and study program behavior. A CI/CD pipeline generates a lot of logging data at every level of the

5. Quick rollback if required

One of the most exclusive benefits of a CI/CD pipeline is that it leads to the quick and easy rollback of code changes if there are any issues in the production environment after a release. If any new code change breaks a feature or general application, you can revert to its previous stable version right away. You can deploy the most recent successful build instantly to avoid production interruptions.

6. Better planning

Organizational designs must be adaptable to changing economic conditions. However, it's difficult for development and testing teams to adapt to rapid changes in dynamic business conditions. A CI/CD
pipeline enables organizations to accomplish this by ensuring that they have a well–organized surplus of items and a continuous line of communication with clients

7. Efficient testing & monitoring

Testing entails automating each test case and experimenting with the program. Any cycle that needs to be repeated over time should be automated, and there are enough innovations available to achieve this goal. Manual testing measures must be evaluated for possible automation outcomes, and in the vast majority of circumstances, there will be ways to automate the equivalent.

The code delivery cycle should create the possibility of running the test suite on each product assembly built without client intervention, resulting in a defined goal of providing quality delivery quickly.

Using continuous monitoring, Ops teams can oversee and ensure that the application is running as expected and that the environment is stable. They must ensure that the applications perform optimally.
This involves the deployment of software that can monitor application health and issues. It may also be necessary for the Ops team to collaborate with the development team to build self-observing or investigation gathering capabilities directly into the applications. Developers have to consider checking the product from start to finish continually

8. Cost-effectiveness

The CI/CD pipeline takes a different approach to software delivery. It can be compared to an assembling unit's delivery pipeline. In any business situation, time and assets are essential. Firms are expected to respond to client demands quickly and effectively with such requirements.
With automated testing hooks at every stage, developers can fix issues early and avoid critical issues in the production environment. With the implementation of a CI/CD pipeline, code quality improves drastically, which, in turn, improves the overall ROI

Takeaway

Businesses looking to improve application performance and quality need a dependable delivery procedure. As the best CI/CD technologies become visible in the marketplace and on DevOps' desktops, we will soon witness a shake-up soon. While Netflix, the poster child for continuous integration and delivery, can complete integration, testing, and delivery in hours, most businesses are far from that efficiency level.

Given the benefits to your organizations, it's a worthwhile goal to pursue. When you combine continuous integration and delivery with your company objectives, you'll be on your way to improving your software delivery timelines and your bottom line.