Deployment Made Easy through CICD

Anytime, Anywhere and Automated.

Definition

CI and CD stands 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.

Why CICD ?

CI/CD allows organizations to ship software quickly and efficiently. CI/CD facilitates an effective process for getting products to market faster than ever before, continuously delivering code into production, and ensuring an ongoing flow of new features and bug fixes via the most efficient delivery method.

## CI vs CDelivery vs CDeployment

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.

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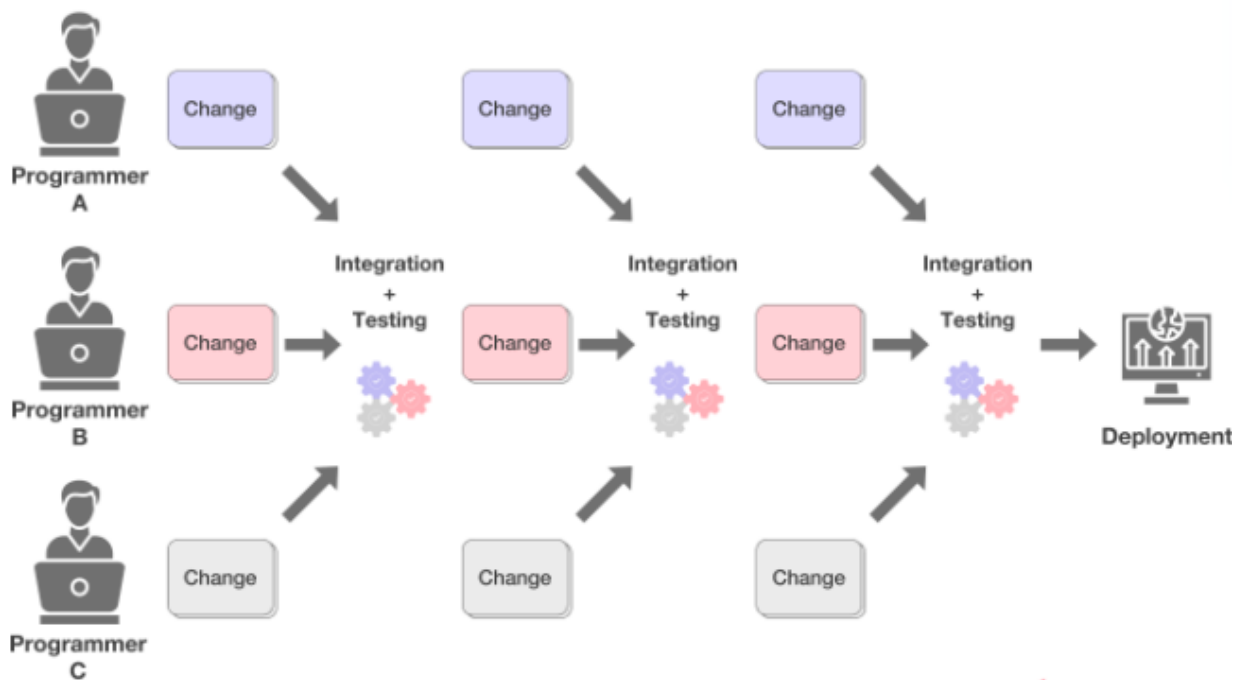
Continuous Deployment is the next step of continuous delivery. Every change that passes the automated tests is automatically placed in production, resulting in many production deployments.

Ways of Working: Traditional vs CICD
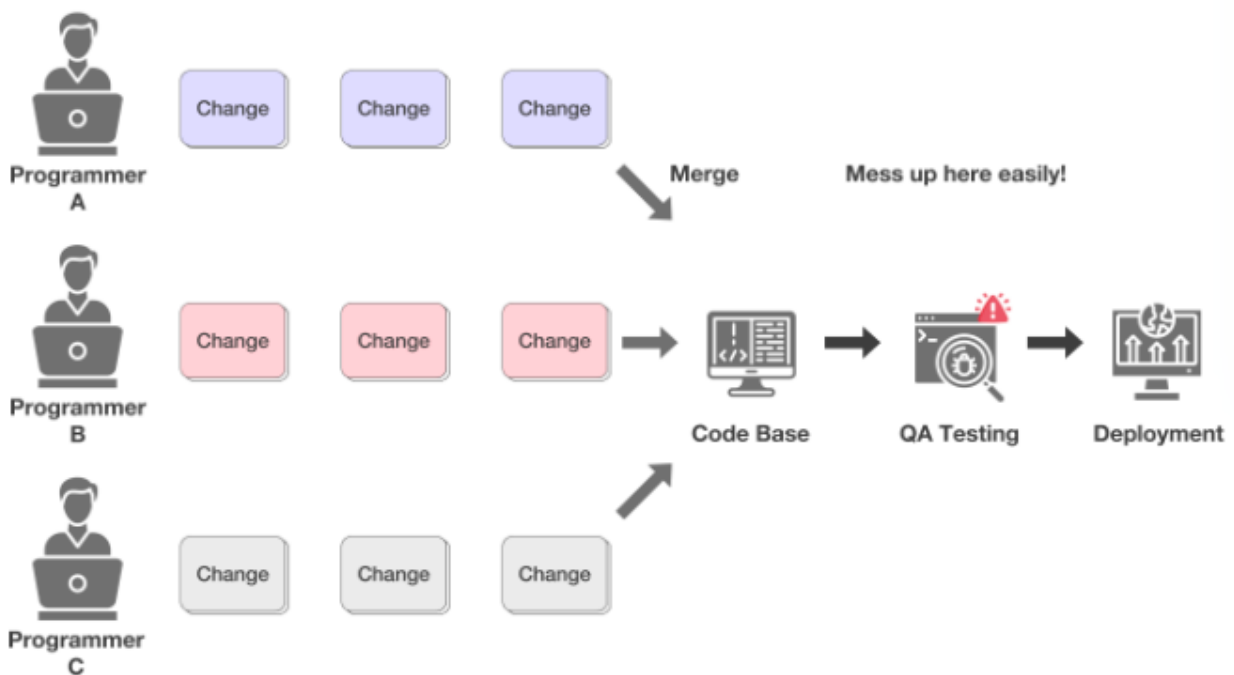
With CI & CD

Key Takeaways

1. Reduce time-to Market

    The goal of a CI/CD pipeline is to build and deliver software to users at a rapid pace. Moreover, software development has gone beyond introducing new features, writing robust code, and understanding users' needs. A CI/CD pipeline enables you to ship changes not just weekly, daily, and even hourly.

2. Effective automation testing

    Understanding your code's performance is vital in a software release; performing it efficiently requires effort and hours. Manual testing is a repetitive process that demands a high concentration level and demands developers to perform the same process with minute variations for the umpteenth time. Therefore, a major aspect of any CI/CD pipeline is to have a set of automated tests that every build must complete.

3. Improved Mean Time to Resolution (MTTR)

With the help of MTTR, development teams can measure the sustainability of repairable features. Also, it establishes the standard time to fix a faulty feature and enables you to determine the time required for failure recovery. CI/CD significantly helps reduce the MTTR, as there are only minor changes in the code and detecting fault isolations is less complicated.

4. Efficient infrastructure

A CI/CD pipeline is largely built on automation that aims to make the release process repeatable and reliable. While building an infrastructure with CI/CD, your initial goal will be to write and run automated tests. After building a strong foundation, your next goal involves automating deployments of your builds to test and staging environments.

5. Accurate progress monitoring

A majority of the CI/CD tools can be used for instrumenting the process and provide a set of metrics that include building, test coverage, failure rates, test fix times, and more. This data can help you recognize areas causing your pipeline to perform slowly and make improvements as and when required.

6. Maximum consumer demand satisfaction

CI/CD helps you adopt and incorporate a customer-first approach. When you release a product, it carefully monitors the initial actions of the customers and maintains a record of the results. As a result, you can study the kind of impression your product creates on the customers.

7. Cloud-based development

A major share of CI/CD's popularity can be attributed to the rise in cloud development since traditional development techniques often prove inefficient with cloud-based applications. It has become the central pillar of streamlining cloud-native development and making the processes faster, consistent, and more efficient.

8. Shorter review cycles and faster debugging

Continuous integration encourages developers to commit their code changes more frequently, at least once daily. Regularly sharing code with the team ensures that every member is building on the same principles, code reviews are quicker, and integrating changes is less time-consuming. Uploading smaller increments also reduces the code reviewer's responsibilities and encourages the entire team to work in collaboration rather than fixing bugs in isolation. This practice also allows remote Superior code quality

One of the most important business benefits of CI/CD is the improvement it brings to the overall quality of the code. Since the development team releases code in small batches, it can be thoroughly tested — for example, by using unit testing, which allows developers to detect and fix the most serious bugs before deploying the software to production.

Moreover, the gradual change applied to the code is more manageable for developers. That's why the chances of finding a critical bug once the product goes live are much lower. Automated testing allows fixing bugs immediately, which is far less costly and easier than fixing them later on. to participate in the process and remove the communication barrier.

Business value and benefits of CICD

❖ Superior code quality

CI/CD helps in the improvement of overall quality of the code. Since the development team releases code in small batches, it can be thoroughly tested — for example, by using unit testing, which allows developers to detect and fix the most serious bugs before deploying the software to production.

Moreover, the gradual change applied to the code is more manageable for developers. That's why the chances of finding a critical bug once the product goes live are much lower. Automated testing allows fixing bugs immediately, which is far less costly and easier than fixing them later on.

❖ Release Software Faster : Shorter Release Time

Since the code changes are small, the development team can **release software builds faster**. Essentially, this could even happen on-demand. This is why CI/CD is so important in improving the team's flexibility and ability to ship functionalities fast.

❖ Cost reduction thanks to automation

The **automation** of the *build*, *test*, and *deploy* phases makes the life of a DevOps team much easier. It also significantly reduces the cost of software development. By automating actions, you can be sure that they're less prone to human error and easier to manage by the team.

❖ Fast feedback | Fail Fast

Another great thing about setting up a CI/Cd pipeline is the availability of instant feedback about the code for a new build. Failure is a normal occurrence, so it's worth applying the principle of failing fast. You can achieve that by automating tasks. When they identify a bug, you can fix it rapidly.

❖ Greater customer satisfaction

Ultimately, the goal of CI/CD is to make customers happy. When dealing with buggy software, your company's reputation is damaged, and customers become irritated. By shipping new features regularly, fixing bugs immediately, and reacting to feedback, you will **delight the end-users** of your application.

❖ Increased business sustainability

Most businesses don't aim to win a sprint — their goal is to win a marathon. Cutting ahead of the competition isn't easy and doing it consistently is even harder. By automating manual tasks, you can prevent your team from burning down all the way before your business becomes successful. A CI/CD pipeline comes at a significant **upfront investment**. However, a well-designed pipeline will position your company on the path to success by innovating faster and meeting customer needs