What is CI/CD

Continuous Integration (CI) allows you to continuously integrate code into a single shared and easy to access repository. Continuous Delivery (CD) allows you to take the code stored in the repository and continuously deliver it to production. CI/CD creates a fast and effective process of getting your product to market before your competition as well as releasing new features and bug fixes to keep your current customers happy.

Benefit of CI/CD

1. Smaller Code Changes

One technical advantage of continuous integration and continuous delivery is that it allows you to integrate small pieces of code at one time. These code changes are simpler and easier to handle than huge chunks of code and as such, have fewer issues that may need to be repaired at a later date.

Using continuous testing, these small pieces can be tested as soon as they are integrated into the code repository, allowing developers to recognize a problem before too much work is completed afterward. This works really well for large development teams who work remotely as well as those in-house as communication between team members can be challenging.

2. Fault Isolations

Fault isolation refers to the practice of designing systems such that when an error occurs, the negative outcomes are limited in scope. Limiting the scope of problems reduces the potential for damage and makes systems easier to maintain.

Designing your system with CI/CD ensures that fault isolations are faster to detect and easier to implement. Fault isolations combine monitoring the system, identifying when the fault occurred, and triggering its location. Thus, the consequences of bugs appearing in the application are limited in scope. Sudden breakdowns and other critical issues can be prevented from occurring with the ability to isolate the problem before it can cause damage to the entire system.

3. Faster Mean Time To Resolution (MTTR)

MTTR measures the maintainability of repairable features and sets the average time to repair a broken feature. Basically, it helps you track the amount of time spent to recover from a failure.

CI/CD reduces the MTTR because the code changes are smaller and fault isolations are easier to detect. One of the most important business risk assurances is to keep failures to a minimum and quickly recover from any failures that do happen. Application monitoring tools are a great way to find and fix failures while also logging the problems to notice trends faster.

4. More Test Reliability

Using CI/CD, test reliability improves due to the bite-size and specific changes introduced to the system, allowing for more accurate positive and negative tests to be conducted. Test reliability within CI/CD can also be considered *Continuous Reliability*. With the continuous merging and releasing of new products and features, knowing that quality was top of mind throughout the entire process assures stakeholders their investment is worthwhile.

5. Faster Release Rate

Failures are detected faster and as such, can be repaired faster, leading to increasing release rates. However, frequent releases are possible only if the code is developed in a continuously moving system.

CI/CD continuously merges codes and continuously deploys them to production after thorough testing, keeping the code in a release-ready state. It's important to have as part of deployment a production environment set up that closely mimics that which end-users will ultimately be using. Containerization is a great method to test the code in a production environment to test only the area that will be affected by the release.

6. Smaller Backlog

Incorporating CI/CD into your organization's development process reduces the number of non-critical defects in your backlog. These small defects are detected prior to production and fixed before being released to end-users.

The benefits of solving non-critical issues ahead-of-time are many. For example, your developers have more time to focus on larger problems or improving the system and your testers can focus less on small problems so they can find larger problems before being released. Another benefit (and perhaps the best one) is keeping your customers happy by preventing them from finding many errors in your product.

7. 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.

8. Increase Team Transparency and Accountability

CI/CD is a great way to get continuous feedback not only from your customers but also from your own team. This increases the transparency of any problems in the team and encourages responsible accountability.

CI is mostly focused on the development team, so the feedback from this part of the pipeline affects build failures, merging problems, architectural setbacks, etc. CD focuses more on getting the product quickly to the end-users to get the much-needed customer feedback. Both CI and CD provide rapid feedback, allowing you to steadily and continuously make your product even better.

9. Reduce Costs

Automation in the CI/CD pipeline reduces the number of errors that can take place in the many repetitive steps of CI and CD. Doing so also frees up developer

time that could be spent on product development as there aren't as many code changes to fix down the road if the error is caught quickly. Another thing to keep in mind: increasing code quality with automation also increases your ROI.

10. Easy Maintenance and Updates

Maintenance and updates are a crucial part of making a great product. However, it's important to note within a CI/CD process to perform maintenance during downtime periods, also known as the non-critical hour. Don't take the system down during peak traffic times to update code changes.

Upsetting customers is one part of the problem, but trying to update changes during this time could also increase deployment issues. Make sure the pipeline runs smoothly by incorporating when to make changes and releases. A great way to ensure maintenance doesn't affect the entire system is to create microservices in your code architecture so that only one area of the system is taken down at one time.