CI/CD AUTOMATION

[UDAPEOPLE]

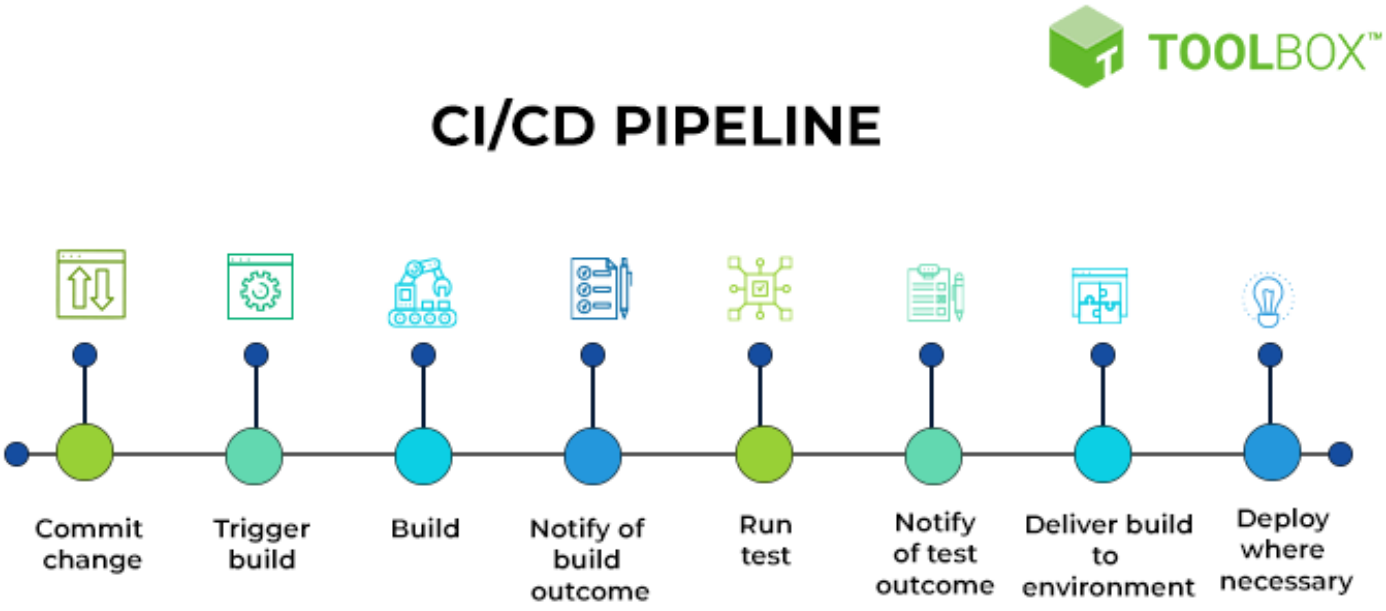
MARCH 14, 2023

WHAT IS CI/CD?

CI/CD, which stands for Continuous Integration and Continuous Delivery, allows us to automate the process of Software tests, builds, and deployments and helps us achieve speed and agility of release. The idea behind Continuous integration, continuous development, and continuous deployment is to support rapid software updates without disturbing system stability and security.

As DevOps practice, CI/CD promotes automation, enabling teams to work in parallel and deliver software faster, securely, and reliably while ensuring zero tolerance for outages.

It's CI/CD behind every organization, whether small or large, able to support its team or customers with quick and reliable support. Here is a long list of benefits of CI/CD that a company experiences based on maturity level.



WHY SHOULD UDAPEOPLE ADOPT CI/CD TO ACHIEVE, BUILD, AND DEPLOY AUTOMATION?

1. Tight Feedback Loop

Rapid feedback is an integral part of the CI/CD. It starts with automated build and test phases to notify you of immediate problems, helping you work more efficiently and effectively than long delays between original work and results.

Likewise, shipping updates regularly give you far more immediate feedback about what you've created than if you changed batches every few months for a major release. You can identify what is working well and prioritize modifications and improvements by collecting feedback, observing user behavior, and tracking key performance indicators.

2. Fast Go To Market

Organizations have taken the lead in transforming their development processes and adopting Agile and DevOps technologies to deliver continuous improvements to their users. Still, the landscape is becoming increasingly competitive to suit many smaller organizations. CI/CD helps all sizes of organizations to deliver software or updates to use frequently.

Understanding the needs of your users, coming up with innovative features, and turning them into robust code is crucial, especially when your competition is doing it better than you. Setting up an automated CI/CD pipeline allows you to push changes even in real-time. Many deployment strategies are there to support the different business use cases. Pursuing changes quickly and confidently means you can respond to new trends and address pain points as they emerge..

3. Collaboration And Communication

CI/CD improves overall communication and accountability among team members. It becomes a common framework for all developers, QAs, and product managers working on a particular project.

For each pipeline running on CI/CD environment, all stakeholders are notified and kept on the same page for any changes and in case of any failure. It allows product owners and developers to effectively communicate about test results and take the desired action based on the severity of the failure.

4. Fault Detection And Isolation

Tracing the root cause of a defect and then pinpointing the exact location of the defect is one of the most stated advantages of CI/CD. CI/CD pipelines are designed to identify bugs or issues as quickly as they are committed to different pipeline stages and don't go beyond until being resolved, limiting the negative consequences of an error by indicating the cause and location.

Such an approach minimizes the consequences of an unresolved or known problem, making the system easier to maintain. CI/CD pipeline as it makes fault isolation simpler and faster before it affects the entire system.

5. Increased Test Reliability

By using CI/CD, you can improve test reliability to a great extent. As specific and atomic changes are introduced into the system, developers or QA can add more relevant positive and negative tests to the changes. This test is also called 'continuous reliability' within the CI/CD pipeline.

It also makes the code more reliable as it often prevents issues caused by working cases on my local system as tests that run locally are executed on a

snapshot of the code on the local machine. Continuous Integration prevents such cases by performing continuous testing on the integrated code.

6. Lower Risk

Developers release code in small batches into a shared repository, enabling them to perform parallel testing. Instead of working in isolation, they often share their builds with the entire team. Teams collaborate to identify critical bugs, ensuring that bad code doesn't make it to production. Thus CI/CD implementation accelerates commercial development by providing high-quality releases that contain fewer errors and bugs.

7. Short Review Time

With Continuous Integration, developers are encouraged to write and submit their code more often – ideally once a day as a rule of thumb. Sharing code regularly with the rest of the team ensures that everyone builds on the same foundation. It supports faster code reviews leading to faster integrations.

These smaller increments of code mean that – as a code reviewer – there's less to spin your head around. As smaller commits generate more specific commit messages, you can see how the logic progresses. And if something needs to change before a commit can be merged, there's less code to rewrite and fewer conflicts to resolve.

8. Efficient Infrastructure

Automation is a key part of any CI/CD pipeline as it automates the releases repeatedly and reliably. Implementing Continuous Integration in the early stages allows infrastructure engineers to focus on automating the build process and writing and running automated tests. Once you have established a solid CI

foundation, the next step is to automate the deployment of your builds for testing and staging environments.

Taking the code as an infrastructure approach involves automating the creation of those environments. Instead of manually managing individual servers, their configurations are scripted and stored under version control to quickly bring new environments online without the risk of accidental changes and inconsistencies.

It makes the continuous delivery phase faster and more robust. Still, it also allows you to quickly respond to requests for additional preview and training environments with minimal interruption to development work.

9. Measurable Progress

Another best advantage of CI/CD is that it provides regular maintenance and updates of the product.
It ensures that release cycles are short and targeted, which blocks fewer features that are not ready for release. In a CI/CD pipeline, maintenance is typically done during non-business hours, saving the entire team valuable time.

In addition, features such as toggle and blue-green deployment enable the smooth and targeted introduction of new product features by upgrading to smaller units of change, which are less disruptive.

10. Cost Reduction

Providing high-quality tailor-made solutions to address business-specific challenges requires a way to meet the increasing time constraints imposed by competitors. Embracing CI/CD is perfect for reducing the time it takes to complete a project and bring new features to the market. The shorter the development cycle, the more likely it meets ambitious time-to-market goals.

11. Improved Mean Time To Resolution (MTTR)

MTTR measures the reliability and maintainability of any or all repairable facilities. It gives you a timeline of the average time it takes to recover from a potential failure.

One of the main advantages of CI/CD is that it helps you reduce this number. Small code changes and quick fault isolation are vital in keeping failures to a minimum. It also helps to recover from any setbacks in no time. The CI/CD pipeline ensures that fixes are quickly tested in integrating the entire code before deploying to production.

12. Maximum Productivity

As we've seen, building a CI/CD pipeline eliminates communication gaps while supporting a leaner, more efficient software development and release process. By using advanced tools to perform repetitive tasks, an automated process also frees individuals from being creative. Instead of following manual test scripts, refreshing the environment, or deploying updates, you can focus on solving problems and experimenting with solutions.

Having room to be more creative and adding value to what you do improves job satisfaction and encourages DevOps to contribute more. This benefits your organization, product, users, and ultimately your bottom line.

13. Easy Way To Produce

Adoption of CI/CD is best done sequentially, starting with CI practices and building your pipeline over time. As you begin to apply changes more frequently, you may start addressing pain points and steps in your current process that wouldn't provide you the expected velocity, such as refreshing data in a test environment or parameters before deploying to a particular machine.

Adding automation to build, test, build environment, and deploy makes each step consistent and repeatable. By breaking it down, you can continue to optimize each step to make your process more efficient. From being a pivotal event that lasts several days to multiple teams, CI/CD matures into a familiar and predictable event with the release.

CONCLUSION

Continuous integration and continuous deployment is a significant part of DevOps as a service that helps to reduce the workload of developers by automating the build and deployment pipeline. The process allows for rapid identification of problems for quick problem-solving. In addition, CICD helps fast-track system update releases and prompt responses to customer feedback.

These are just a few of the benefits of CI/CD. Integrating CI/CD into your Udapeople software development pipeline will only benefit this business. Continuous integration is not just a tool for continuous delivery of software development. It's a fundamental approach that can set your business apart as a leader in your field.