The benefits of CI/CD

The fundamentals and benefits of CI/CD to achieve, build, and deploy automation for cloud-based software products.

Firstly, we will talk about business before CI/CD:

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.

Secondary, 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.

CI/CD automation can eliminate manual tasks, prevent coding errors, and 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.

CI/CD enables benefits like scalability, elasticity, and improved performance characteristic of cloud-native applications.

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.

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.

To increase revenue, A CI/CD pipeline is like a turbo engine when it comes to accelerating the code deployment pace and time-to-market of the final product. It enables you to ship code changes not just every week, but every day and even hourly. As testing is automated, the code is automatically deployed if it meets the predefined criteria. This enables you to release software to production several times at a rapid pace while keeping quality intact. Moreover, the automation the CI/CD pipeline brings to the software development lifecycle reduces manual labor and time required for creating and maintaining deployment scripts and tools.

The developers would easily know there was an issue with code, but they used to struggle to know where exactly the problem was happening. Now, the automated testing practices of the CI/CD pipeline improved the visibility across the software development lifecycle. Developers can easily spot and isolate code issues. This, in turn, significantly improved productivity.

To Reduce Costs, after Merge we can catch compile errors because less developer time on issues from new developer code and automate infrastructure cleanup because less infrastructure costs from unused resources.

CI/CD pipeline reduces human intervention across the DevOps lifecycle by automating the handoffs, version controlling, source code management, deployment processes, and testing, among others. This significantly saves the time and money required to develop and deliver high-quality software. Moreover, with a successful CI/CD pipeline in play, the development teams aren't plagued with endless 'code fix' requests, so they can keenly focus on the next projects, maximizing the overall ROI for the company.

To Avoid Costs, Code testing and debugging is arguably the most important aspect of software development. And testing is the prime reason the releases are delayed. The delayed releases not only impact the time-to-market, but they also affect the business from a cost, branding, and reputation perspective. Before DevOps and CI/CD, the software testing and debugging were conducted manually, a time-consuming process. But, with CI/CD pipeline, you can significantly cut down on manual testing and debugging, dramatically improving software delivery.

Also, a successful CI/CD pipeline enables your DevOps teams to continuously integrate small batches of code instead of the entire application. This approach helps the developers to easily identify the anomalies and fix them. So, you can avoid significant outages and other key issues by flagging bugs and vulnerabilities before they make it to production and disrupt the entire application.

To protect revenue, we can use automated smoke tests that is reduced downtime from a deploy-related crash or major bug.

Smoke testing plays an important role in software development as it ensures the correctness of the system in initial stages. By this, we can save test effort. As a result, smoke tests bring the system to a good state. Once we complete smoke testing then only we start functional testing.

- All the show stoppers in the build will get identified by performing smoke testing.
- Smoke testing is done after the build is released to QA. With the help of smoke testing, most of the defects are identified at initial stages of software development.
- With smoke testing, we simplify the detection and correction of major defects.
- By smoke testing, QA team can find defects to the application functionality that may have surfaced by the new code.
- Smoke testing finds the major severity defects.
- Easy rollback is one of the key benefits offered by CI/CD. A CI/CD pipeline empowers development teams to fail fast and recover even faster. Simply put, the CI/CD pipeline enables your developers to easily push code into production and, if any issues arise, simply roll it back. This ultimate ability to roll back code saves time, resources, and expenses by helping teams to fix the problem code at a faster pace.