A STUDY AND ANALYSIS OF CONTINUOUS DELIVERY, CONTINUOUS INTEGRATION IN SOFTWARE DEVELOPMENT ENVIRONMENT

Yasmine SKA
Assistant Professor
Department of Computer Science
Auxilium College(Autonomous)

Janani P
Assistant Professor
Department of Computer Science
Auxilium College(Autonomous)

ABSTRACT

This paper explains the features and benefits of using continuous delivery, continuous integration (CI/CD). Also investigates impacts of (CD/CI). Continuous delivery(CD) is a set of practices designed to optimize the process of taking changes from version control to production or release to manufacturing. Key elements include comprehensive use of version control, automation of the test and deployment process and the application of continuous integration to rapidly validate the correctness of every change through running the automated build and test process. The goal of Continuous Delivery(CD) is to find ways to deliver high-quality, valuable software in an efficient, fast, and reliable manner. Continuous integration (CI) is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. The key goals of CI are to find and address bugs more quickly, improve software quality, and reduce the time it takes to validate and release new software updates. Continuous integration focuses on smaller commits and smaller code changes to integrate. A developer commits code at regular intervals, at minimum once a day. The developer pulls code from the code repository to ensure the code on the local host is merged before pushing to the build server. At this stage the build server runs the various tests and either accepts or rejects the code commit. The basic challenges of implementing CI include more frequent commits to the common codebase, maintaining a single source code repository, automating builds, and automating testing.

CI and CD provides an ideal scenario for the organization's application teams. When CI/CD is used code quality is improved, and software updates are delivered quickly and with high confidence that there will be no breaking changes. CI/CD is a practice that enables rapid software changes while maintaining system stability and security.

1. INTRODUCTION

Enterprises today face the challenge of rapidly changing competitive landscapes, evolving security requirements, and performance scalability. Enterprises must bridge the gap between operations stability and

rapid feature development. Continuous integration and continuous delivery (CI/CD) is a practice that enables rapid software changes while maintaining system stability and security.

Modern organizations that treat software development as a competitive advantage face a problem: they need to deliver software ever faster while preserving the quality and stability of the systems. A set of technical practices has evolved to meet this need, which together have been popularized under the term continuous delivery (CD).

2. CONTINUOUS DELIVERY

Continuous delivery is typically comprised of a few key processes. In one, developers break down their work into small updates or changes that are made on trunk or mainline in version control. Every time the developers commit a change, they get rapid feedback within minutes from automated tests, another key aspect. Automated testing allows for fast feedback and learning, providing developers an opportunity to fix any problems immediately when the automated tests fail. Together, these practices are known as continuous integration. When the build and tests pass, downstream processes such as comprehensive automated acceptance testing, manual exploratory testing, and performance tests are triggered. These rely on the ability to deploy the software automatically to an environment created automatically from system and application configuration information stored in version control (known as deployment automation).

The goal is to take activities such as integration and comprehensive testing that are traditionally performed after a release is "dev complete," and build them into the software development process. This is intended to remove the need for stabilization and hardening phases.

CI is a widely established development practice in software development industry, in which members of a team integrate and merge development work (e.g., code) frequently, for example multiple times per day. CI enables software companies to have shorter and frequent release cycle, improve software quality, and increase their teams' productivity. This practice includes automated software building and testing.

3. THE VALUE OF CONTINUOUS DELIVERY

Continuous Delivery is important to Enterprise , Because it enables the business to

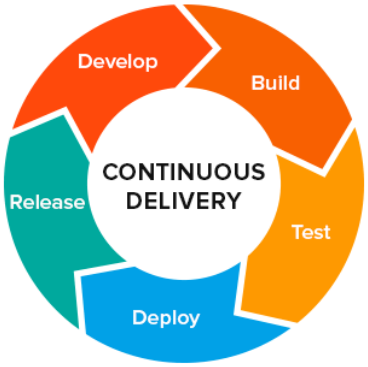
rapidly respond to the expectations of their customers while improving the quality of their products at a lower cost.

The goal of Continuous Delivery is to find ways to deliver high-quality, valuable software in an efficient, fast, and reliable manner. Continuous Delivery is about market speed and moving quickly from whiteboard to rollout, faster than your competition. This market speed means a shorter feedback loop and a faster time to value. With a shorter feedback loop, you fail faster, fix faster, adjust faster, and succeed faster. This agility provides a distinct business competitive advantage which is why companies like Amazon and Netflix are dominating their respective competitive landscape, transitioning into adjacent markets, and taking market leadership. These companies can adapt faster to the changing marketplace, and deliver new features and services faster than their competitors. The central pattern of continuous delivery is the deployment pipeline which is an automated implementation of application's build, deploy, test, and release process. The deployment pipeline is instantiated whenever a change is made to an application.

While the deployment pipeline is an effective pattern for getting software from development to release, mapping an automated deployment pipeline across an entire enterprise can be challenging. Organizations tend to start the journey by taking a more incremental approach, starting with development and build processes that terminate with Continuous Integration, which is the foundation of the deployment pipeline.

The vary environment, system dependencies cause a problem and new bugs arise while building the system which became difficult to maintain. Some problems people face into the process as, maintaining different environments, package versions, repositories, rollbacks, etc.

Developers used to throw their code over the metaphorical wall, and operators were responsible for keeping that code running in production. Developers were concerned with shipping code, while operators were concerned with reliability. This misalignment is filled by the continuous DevOps process.



CONTINUOUS DELIVERY PHASES

As shown in the above diagram, Continuous Delivery process consist of the SDLC phases with automation at each phase. With this user can deliver the changes to customers quickly in a reliable way and achieve continuous end-to-end automation i.e. one click deployment, quick release, etc. So, no specific schedule is needed to deploy new things. And, user could easily manage, debug and deploy micro-services. Thus code must be robust, which should cover all the edge test cases which will make the standardized system workflow. This solves the complex problems at deploying stage and makes the process seamless. User can release more often, thus accelerating the feedback loop with customers.

The continuous delivery automates the entire software delivery process. Every commit will trigger the automated build, test and deploy to the respective environments. The deployment to production can be manual to validate the desired state while pushing to production. It focuses on reducing the application lifecycle while delivering values faster with best strategy to production with-in a quick succession.

4. BENEFITS OF CONTINUOUS DELIVERY

CD provides numerous benefits for your software development team including automating the process, improving developer productivity, improving code quality, and delivering updates to your customer

Automate the Software Release Process

CD provides a method for your team to check in code that is automatically built, tested, and prepared for release to production so that your software delivery is efficient, resilient, rapid, and secure.

Improve Developer Productivity

CD practices help your team's productivity by freeing developers from manual tasks, untangling complex dependencies, and returning focus to delivering new features in software. Instead of integrating their code with other parts of the business and spending cycles on how to deploy this code to a platform, developers can focus on coding logic that delivers the features you need.

Improve Code Quality

CD can help you discover and address bugs early in the delivery process before they grow into larger problems later. Your team can easily perform additional types of code tests because the entire process has been automated. With the discipline of more testing more frequently, teams can iterate faster with immediate feedback on the impact of changes. This enables teams to drive quality code with a high assurance of stability and security. Developers will know through immediate feedback whether the new code works and whether any breaking changes or bugs were introduced. Mistakes caught early on in the development process are the easiest to fix.

Deliver Updates Faster

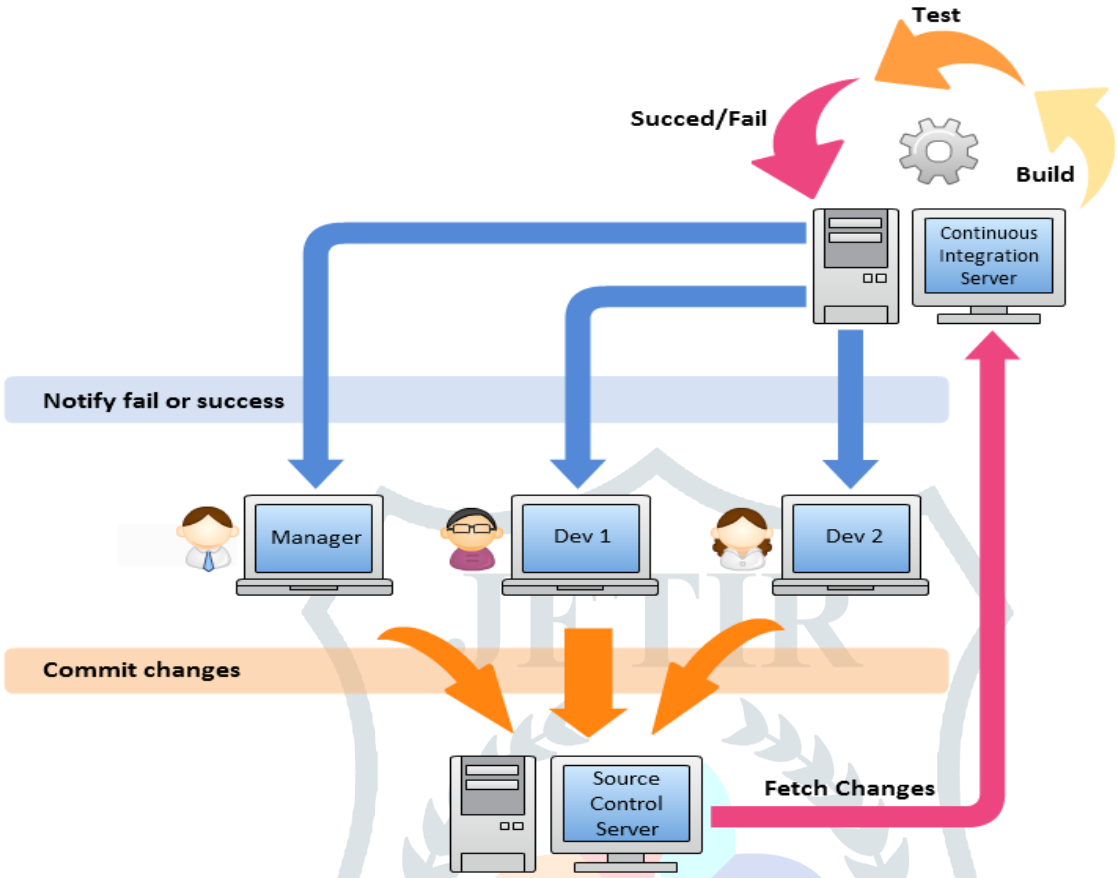
CD helps your team deliver updates to customers quickly and frequently. When CI/CD is implemented, the velocity of the entire team, including the release of features and bug fixes, is increased. Enterprises can respond faster to market changes, security challenges, customer needs, and cost pressures. For example, if a new security feature is required, the team can implement CI/CD with automated testing to introduce the fix quickly and reliably to production systems with high confidence. What used to take weeks and months can now be done in days or even hours.

5. CONTINUOUS INTEGRATION

Continuous Integration (CI) is a widely established development practice in software development industry , in which members of a team integrate and merge development work (e.g., code) frequently, for example multiple times per day. CI enables software companies to have shorter and frequent release cycle, improve software quality, and increase their teams‘ productivity . This practice includes automated software building and testing. It is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. CI most often refers to the build or integration stage of the software release process and requires both an automation component (e.g., a CI or build service) and a cultural component (e.g., learning to integrate frequently).

The key goals of CI are to find and address bugs more quickly, improve software quality, and reduce the time it takes to validate and release new software updates. Continuous integration focuses on smaller commits and smaller code changes to integrate. A developer commits code at regular intervals, at minimum once a day. The developer pulls code from the code repository to ensure the code on the local host is merged before pushing to the build server. At this stage the build server runs the various tests and either accepts or rejects the code commit. The basic challenges of implementing CI include more frequent commits to the common codebase, maintaining a single source code repository, automating builds, and automating testing. Additional challenges include testing in similar environments to production, providing visibility of the process to the team, and allowing developers to easily obtain any version of the application.

As integrating takes place so frequently, there is significantly less back-tracking to discover where things went wrong, so user can spend more time building features.



CONTINUOUS INTEGRATION LIFECYCLE

Continuous Integration is cheap. Not integrating continuously is expensive. If it is not followed in continuous approach, then it will have longer periods between integrations. This makes it exponentially more difficult to find and fix problems. Such integration problems can easily knock a project off-schedule, or cause it to fail altogether.

6. BENEFITS OF USING CONTINUOUS INTEGRATION

Continuous Integration brings multiple benefits to your organization:

- Using the CI is beneficial for many reasons.
- **Reduced integration risk.** More often than not, working on projects means multiple people are working on the separate tasks or parts of the code. The more people, the riskier the integration. Depending on how bad the problem really is, debugging and solving the issue can be really painful and can potentially mean a lot of changes to the code. Integrating on a daily basis or even more frequently can help reduce these kinds of problems to a minimum.
- **Higher code quality.** Not having to worry about the problems, and focusing more on the functionality of the code results in a higher quality product.

- **The code in version control works.** If you commit something that breaks the build, you and your team get the notice immediately and the problem is fixed before anyone else pulls the "broken" code.
- **Reduced friction between team members.** Having the impartial system in place reduces the frequency of quarrels between team members.
- **The quality of life improvement for testers.** Having different versions and builds of the code can help isolate and trace bugs efficiently, and it makes life easier for the QA team.
- **Less time deploying.** Deploying projects can be very tedious and time-consuming, and automating that process makes perfect sense.
- **Increased confidence and morale.** People that don't work for fear of breaking something, are more likely to produce better results and can focus their energy and concentration on producing instead of worrying about potential consequences of their actions.
- One side effect of all these benefits is that new team members will have a much easier time getting into the project. Having a clear vision of the building process can greatly speed up adaptation of the newest dev on the team.

7. DEVEOPS AND AUTOMATION

Automation forms the core practice of DevOps, wherein a special focus is laid on setup, configuration, deployment and assisting infrastructure and applications. Automation helps set up environments more rapidly in a standardized and automated manner. Earlier, server configuration an d application deployment were predominantly manual processes with high vulnerability to errors, unreliability and inability to support agile business. To

address these concerns, organizations were employing highly-skilled resources to provide manual configuration. But this did not solve the problem leaving scope for impact on critical, high-value activities such as software release, ma chin e configuration, operating system patching, troubleshooting or bug fixing, within a business. This is where Automation comes as a savior, automating majority of the critical business tasks! DevOps environment presents a higher level end-to-end automated process, eliminating enterprise burden caused by manual intervention or access to the production environments. DevOps relies on tools to automate large parts of the end-to-end software development and deployment process.

Key DevOps Benefits - A Snapshot

1. Ensures stable and reliable operating environments

2. Facilitates continuous release and deployment

3. Improves quality and time for innovation

4. Treats infrastructure as code

5. Ensures smaller and faster deployments, ensuring faster time-to-market

6. Improves ROI, the key to any business success

7. Mitigates risk by reducing time to delivery

8. Identifies problems and provides smooth and effective resolution.

8. DEVOPS TOOLS FOR CONTINUOUS DELIVERY, CONTINUOUS INTEGRATION

Fabric is a Python (2.5-2.7) library and command-line tool for streamlining the use of SSH for application deployment or systems administration tasks. It provides a basic suite of operations for executing local or remote shell commands and uploading/ downloading files as well as performing auxiliary functionality such as prompting the running user for input or aborting execution.

Capistrano is a remote multi-server automation and deployment tool written in Ruby. This tool extends the Rake DSL with methods specific to running commands on servers.

Jenkins is a continuous integration and continuous delivery platform that makes it easy to continuously build and test software projects. It improves productivity by allowing developers to easily integrate changes and also helps users to access f resh builds without any hassles. Jenkins also integrates with a large number of testing and deployment technologies and provides ways to def ine your build pipelines to ensure continuous delivery of software projects.

Through CI or Continuous Integration and CD or Continuous Delivery pipeline. This way, you can reduce the tension centered during the period of product release. The product deficiencies are quickly analyzed so the product pipeline becomes smoother and more frequent, giving your customers updated and highly qualitative products.

Both CI and CD play a very crucial role in automating the software delivery process. When code changes are made or when the developer checks the code into the source repository, those changes will be instantly reflected in the product delivery pipeline. Through Continuous Integration, integration problems are taken care of, and the developer doesn't have to worry about software issues.

And Continuous Delivery is a system where the development team releases a product predictably and frequently, but it is done automatically. In the traditional manner, you have the developers at one end of the

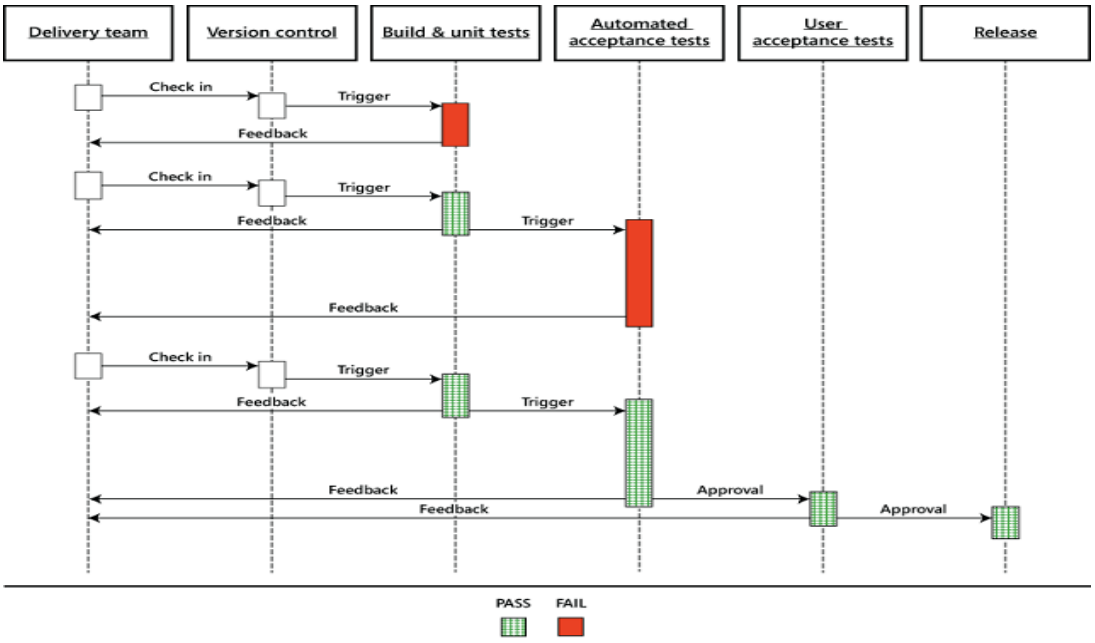
spectrum, i.e. the starting area of the spectrum. But at the receiving end, there are operations personnel, and the product eventually goes through delays at every hand-off leading to frustrated customers and disgruntled teams. This process was tedious and completely error-prone, and of course, often experienced unforeseen delays as well. These problems can be completely eliminated through CD.

Continuous Delivery, an extension of Continuous Integration, is done through Continuous Testing. This technique allows for automated testing that is integrated with software delivery pipeline, leading to validation of every change flowing through it. This way, businesses ensure that the product they deploy will always be usable to the consumers. The delivery procedure is also automated, making the whole process rapid and error-free, and in selected infrastructure environments. Through CI and CD, it is possible to deploy apps in stores with impeccable speed and frequency. The article discusses how to incorporate CI and CD on an existing delivery model, and how this can be done seamlessly.

9. PLANNING FOR CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY

The adoption of CI/CD has really changed the way developers and testers deploy and ship products. The growth of software releases has been tremendous - from Waterfall to Agile and DevOps. It is through DevOps that you have Continuous Integration, Continuous Delivery, (CI/CD) and Continuous Deployment. Through these methods, continuous deployment is done on a monthly, quarterly or bi-annual basis, and sometimes weekly too.

Given below is a typical CI/CD workflow.



Steps to be taken while planning to implement the CI/CD workflow:

i. Weigh the benefits

By adopting CI/CD, you can enjoy delivering faster results. The response to market demands and changes can be met with equal diligence. This means when new features are developed and created, they can be instantly deployed; no waiting time involved.

Productivity levels reach an all-time high because all the repetitive tasks are automated and then deployed. The pipelines are also automated, thanks to the integration of an impressive collection of test automation tools. As manual work is considerably less, the team members can focus on strategic work that provides more value to the company. It is possible to arrange for a convenient delivery time, making it flexible and less stressful.

ii. Consider the requirements

Once you weigh the benefits enjoyed with CI/CD delivery, you can go through the requirements of the project. This would include analyzing the budget, examining areas of allotment and other possibilities.

The team is another major factor. Do you have the right people in your team to proactively resolve issues? Consider the team's development capabilities as they should be proactive rather than reactive.

The success of the team can be ensured when the team leaders are competent enough and understand the full potential of CI/CD and why it is important to embrace this trend.

iii. Define Key Performance Indicators (KPIs)

KPIs are important to monitor progress and analyze continuous improvements. Here are some KPIs in CD/CI delivery model.

Stability index: If the Key Performance Indicators tend to steer towards any particular direction, you need to have other such performance indicators to balance it. This is where stability index comes in.

iv. Code quality index

The company enjoys a key competitive advantage when they can take code changes quickly and easily. Implementing DevOps in the team can help ship code about 30 times faster and complete deployments 8,000 times faster, according to a survey by Puppets Labs. The code quality index is where the different viewpoints will be factored in. The real measures of code quality can report code duplication, memory leaks and security vulnerabilities.

v. Map business KPIs to technical KPIs

The Sprint is another value that business focus on. In order to ensure positive business results, businesses measure the tests performed in that particular sprint, and sometimes the total number of tests in the sprints. Each sprint should contribute something to the tests. For example, there should be a rise in the number of customers after the last release, or a rise in the advertisement clicks in the previous quarter.

10. CONCLUSION

Continuous Integration and Continuous Delivery provides an ideal scenario for organization's application teams. Developers simply push code to a repository. This code will be integrated, tested, deployed, tested again, merged with infrastructure, go through security and quality reviews, and be ready to deploy with extremely high confidence. When CI/CD is used code quality is improved, and software updates are delivered quickly and with high confidence that there will be no breaking changes. The impact of any release can be correlated with data from production and operations. It can be used for planning the next cycle, too—a vital DevOps practice in the organization's cloud transformation..

11. REFERENCES

[1] A. Phillips, M. Sens, A. de Jonge, and M. van Holsteijn, The IT Manager's Guide to Continuous Delivery: Delivering business value in hours, not months: XebiaLabs, 2015. [2] J. Humble, and D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, 1st edition ed.: Addison-Wesley Professional, 2010.

[3] B. Fitzgerald, and K.-J. Stol, ―Continuous Software Engineering: A Roadmap and Agenda,‖ Journal of Systems and Software, vol. 123, 2017.

[4] M. Leppanen, S. Makinen, M. Pagels, V.-P. Eloranta, J. Itkonen, M. V. Mantyla, and T. Mannisto, ―The Highways and Country Roads to Continuous Deployment,‖ IEEE Software, vol. 32, no. 2, pp. 64-72, 2015.

[5] L. Chen, ―Continuous Delivery: Huge Benefits, but Challenges Too,‖ IEEE Software, vol. 32, no. 2, pp. 50-54, 2015.

[6] A. A. U. Rahman, E. Helms, L. Williams, and C. Parnin, ―Synthesizing Continuous Deployment Practices Used in Software Development,‖ in Agile Conference (AGILE), 2015, pp. 1-10.

[7] H. H. Olsson, H. Alahyari, and J. Bosch, ―Climbing the "Stairway to Heaven" -- A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software ‖ in 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), 2012, pp. 392-399.

[8] P. Rodríguez, A. Haghighatkhah, L. E. Lwakatare, S. Teppola, T. Suomalainen, J. Eskeli, T. Karvonen, P. Kuvaja, J. M. Verner, and M. Oivo, ―Continuous deployment of software intensive products and services: A systematic mapping study,‖ Journal of Systems and Software, vol. 123, pp. 263-291, 2017.

[9] Youssef Bassil. "A Simulation Model for the Waterfall Software Development Life Cy- cle", International Journal of Engineering & Technology (iJET), ISSN: 2049-3444, Vol. 2, No. 5, 2012 http://iet-journals.org/archive/2012/may_vol_2_no_5/255895133318216.pdf

[10]. D. Stahl and J.Bosch. "Modelling continuous integration practice differences in industry software developent", Journal of Systems and Software, vol. 87,

pp. 48-59, 2014.