UDAPEOPLE CI/CD PROJECT PROPOSAL

INTRODUCTION

Just like in software engineering, whenever there is any new implementation, it must go through cultural discourse to figure out the pros and cons associated with its adoption. The same applies to the adoption of CI/CD in the software development processes. Many times, developers and management have a misunderstanding about the implementation of CI/CD

Continuous integration, delivery, and deployment are DevOps practices. In other words, they are techniques that implement the DevOps ideals.

To understand the DevOps concept, we connote that to go more further than the Agile software methodology where even though the developers might be working more efficiently within their own team once the build was handed over to operations to deploy staging the process would often bottleneck. Missing dependencies, environment configuration issues, and bugs that could not be replicated on developers' local machines created endless back and forth between teams and disagreements over which side was responsible for fixing the problem. This is where the DevOps concepts came alive.

What is Continuous Integration?

Continuous Integration is the practice of running the steps that were traditionally performed during “integration” little and often throughout the development process, rather than waiting until code is complete before bringing it all together and testing it.

For example, assuming you have more than one person working on a software product – which is the norm for most commercial and open-source software – at some point, you need to combine the pieces each developer has worked on and check that the result works as intended. With continuous integration that point happens at least once a day, if not more.

With continuous integration, your developers share their projects changes by committing them to source control regularly (like GitHub, Bitbucket, etc.) – at least once a day – and check that the solution builds and passes tests. This means that if something breaks, there are far fewer changes to go through to find the source of the problem. Getting feedback quickly also makes it easier to fix any issues, because you haven’t lost the context of what you were doing.

What is Continuous Delivery?

Continuous Delivery builds on the foundations of build and test automate established with continuous integration. With continuous delivery, if a build passes all previous stages in the pipeline successfully it is automatically released to production. This means that as soon as any change to your software has passed all tests it is delivered to your users.

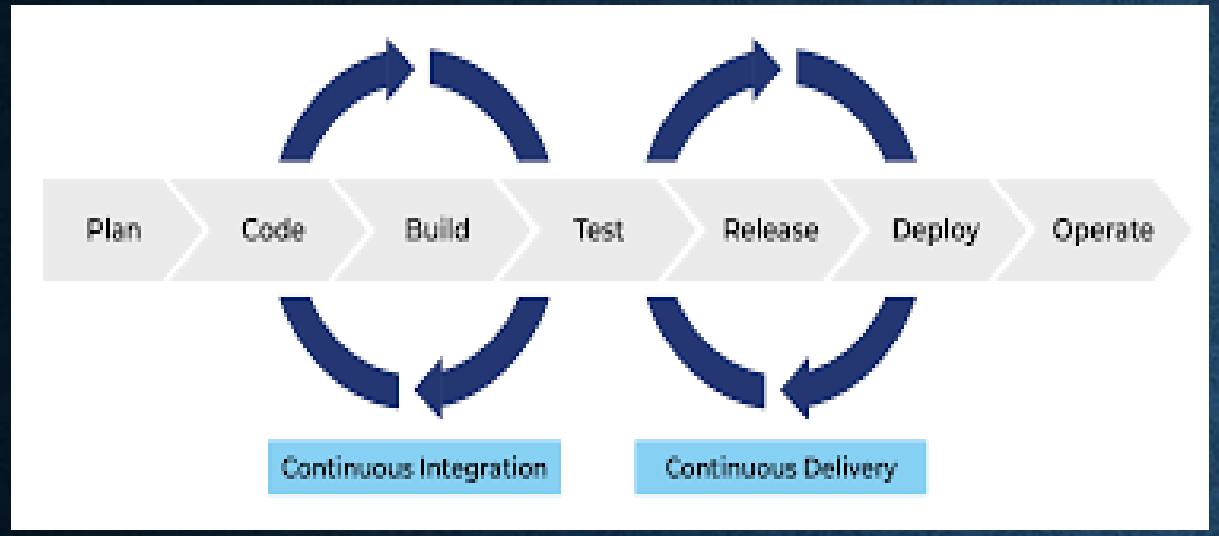
What is CI/CD Pipeline?

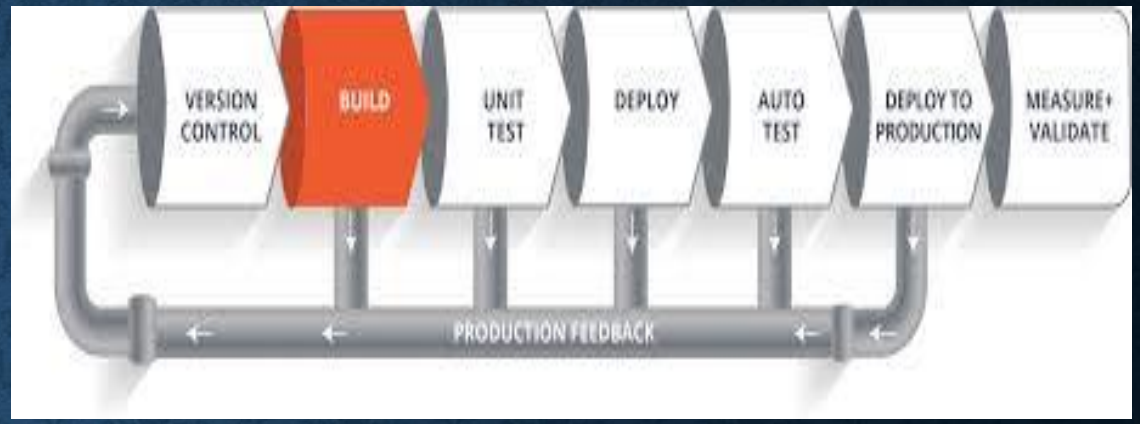
When we talk about a CI/CD pipeline, we're referring to the series of steps that your code goes through in order to get from your development machine, through testing and staging, and finally out of the door and into your users' hands.

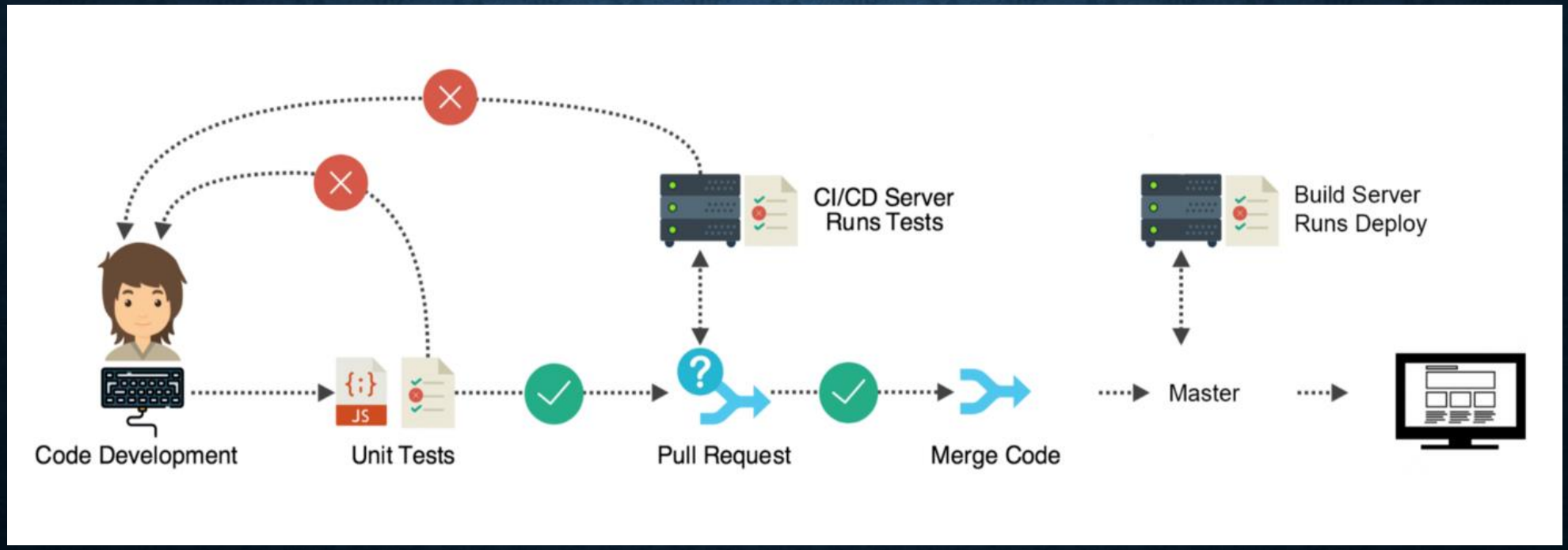
As the CI/CD strategy is to execute this process on a very regular basis – usually multiple times a day – it's essential to automate as much of it as possible, with each step either triggering the next one or raising a flag if something has gone wrong.

Automation not only speeds up the overall process, and therefore the individual feedback loops, but also ensures each step is performed consistently and reliably.

Below is a diagrammatic representation of the CI/CD pipeline.







BENEFITS OF CI/CD PIPELINE

Some of the numerous benefits of CI/CD includes:

- **Cost Reduction**

Anytime a human does not have to intervene in the software development process, time, and thus money, are saved. That is why automation is the underpinning to successful DevOps practices. CI/CD automates the handoffs, the source code management, the version control system, the deployment mechanisms, and, of course, so much of the testing. On the other hand, it saves costs on maintenance and upgrades, and eliminate unnecessary capital expenditure

- **Reduced time-to-market:**

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.

Therefore, it is possible to launch new features faster and implement deployment strategies to help collect feedback that can be incorporated promptly into the upcoming update.

BENEFITS OF CI/CD CONTD.

- **Maximum users 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. Moreover, it lets you keep your product up to date with constant checks for new updates or minor changes. These factors contribute to a high level of user satisfaction.

- **Unites teams for faster product shipments:**

Over the last decade, development teams invested in agile and began developing quicker and quicker. However, since this happened in isolation, operations teams have struggled to keep up and cannot release software at the same pace. DevOps unites these teams and ships software faster in that other teams, like operations, get to share the benefit of working in an agile or iterative environment

- **Improved Average Time to Resolution:**

With CI/CD, 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

CONCLUSION

To remain competitive in today's environment, you need to move faster—and with more precision—than your competitors. DevOps enables that by helping your teams focus on the customer experience, uniting teams for faster product shipments, simplifying the goals of each release, introducing automation (which reduces errors and frees developer time for other projects) and creating a feedback loop that benefits the entire company

THANK YOU!