UdaPeople

Proposal for an organization to adopt CI/CD

Introduction to CI/CD

- CI/CD which stands for Continuous Integration and Continuous Delivery is a software engineering practice which is aimed at creating an environment to automate the process of building, testing, analyzing and monitoring code changes that is being pushed to production. This help developers and organizations to develop, test and release software in quick and consistent fashion.
- This practice is interestingly divided into three(3) categories which all rely on one another. These are: continuous integration, continuous delivery and deployment. Below is the breakdown of the reasons why organizations should adopt CI/CD.

Benefits of CI/CD

- Makes it easy to collaborate on projects. Team members can work from various locations and still have the same impact as if working in the same premises. They are able to push code changes to a remote repository and merge those changes to production thereby making work faster and saving time.

- Unlike the traditional software release process which is premises based and updates delivered irregularly, CI/CD ensures that software release process is automated, code tested and updates delivered frequently to ensure that software do not go offline and keep customers and clients unhappy. This helps to reduce the cost of loosing customers to competitors.

- CI/CD ensures better code quality. Testing a code’s behavior is an essential aspect of software development, however doing it thoroughly can be time consuming. A very important aspect of CI/CD is the series of automated tests that are run on every build. Though writing this tests require a lot of time, doing so pays a lot of dividends. Compared to the manual testing, automated testing is more reliable and quicker to run. This continuous testing ensures bugs discovered per build are fixed and in the end ensures better code quality.

- Continuous deployment also allows organizations to benefit from consistent early feedback. Features can immediately be made available to users and defects or unhelpful implementations can be caught early before the team devotes extensive effort in an unproductive direction. Getting fast feedback that a feature isn’t helpful lets the team shift focus rather than sinking more energy into an area with minimal impact

Small, Iterative Changes

- The practice of CI/CD encourages small changes. Developers are encouraged to break major tasks into smaller ones and commit them early by depending on a special technique known as branching by abstraction. This ensures that new implementations are built outside of the main branch. That is committing those small changes to a branch other than the main. This helps to protect the main branch from in-progress changes.

- CI/CD also ensure efficiency in infrastructure provisioning. Automation forms the central point of CI/CD, serving to make release process repeatable and reliable. Once CI is done, the next phase is deployment of your builds to test and staging environment. Taking infrastructure as code approach involves automating the creation of those environments. This is accomplished by scripting and storing the configuration in a version control so that new environments can be brought up without the risk of inadvertent changes and inconsistencies. This makes continuous delivery stage faster and robust.