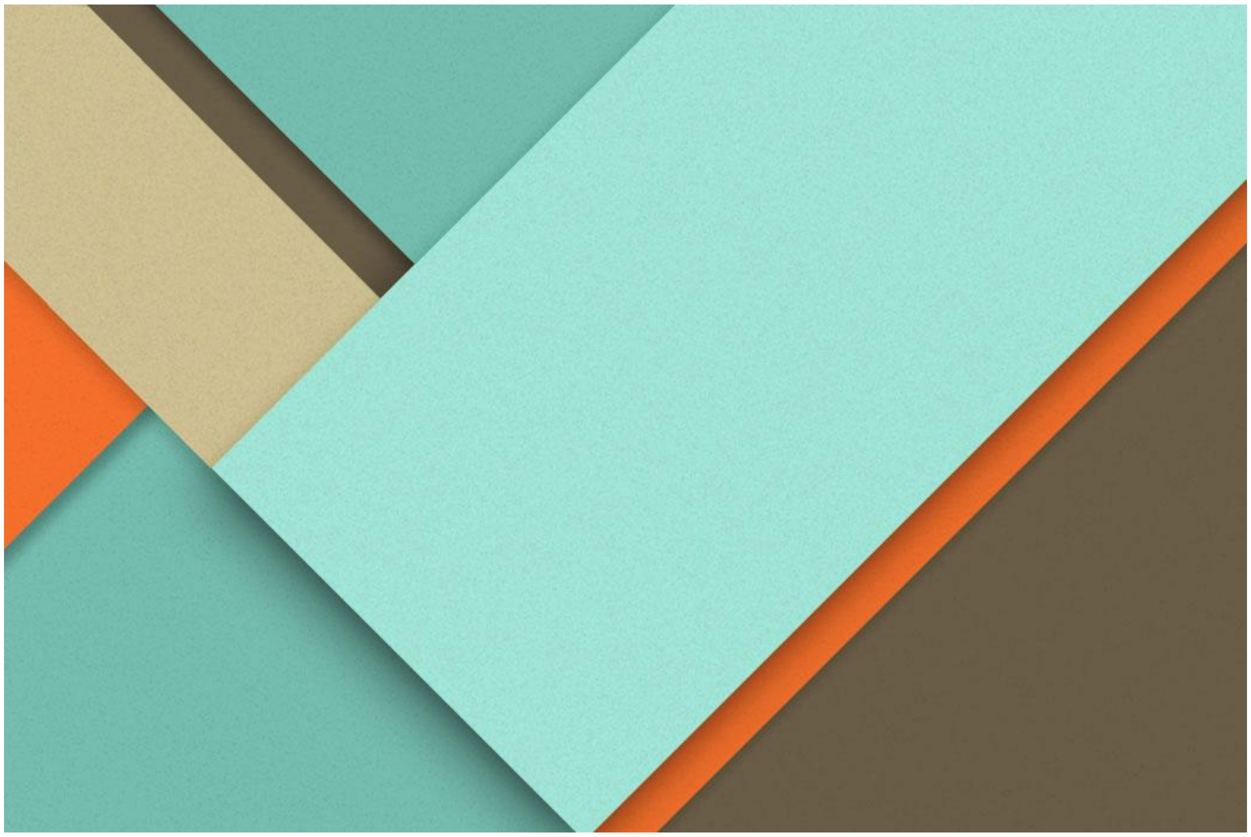

CI/CD Proposal

—

Shamsuddeen Abdulkadir
UdaPeople

Overview

This Document presents the Implementation of Continuous Integration and Continuous Delivery(CI/CD) in relation to the value it adds to the business world by expanding, protecting revenue, controlling costs, or reducing costs in the **UdaPeople** product development process.

Goals

1. Improve overall software release by adopting DevOps methodologies.
2. Automation of manual processes all through the Development Lifecycle.
3. Manage Costs and Improve Revenue Generation.

Introduction to CI/CD

CI/CD is basically a method of frequent delivery of Software Applications to customers by introducing automation of processes in the Software Development phases. The main concept behind CI/CD is Continuous Integration, Deployment and Delivery. These are solutions to problems attributed to or that occur during integration of new code. It is known as; ***Integration Hell.***

This concept introduces automation and continuous monitoring throughout the lifecycle of Applications, from integration and testing phases to delivery and deployment. Put together, these connected practices are also referred to as *CI/CD Pipeline.*

The acronym CI/CD has different meanings. The "CI" in CI/CD is often referred to as continuous integration, which is the automation process for software developers. A Successful CI basically means new code added to an app is regularly built, tested, and merged to a shared repository. It's a solution to the problem of having too many branches of an app in development at once that might conflict with each other.

The "CD" in CI/CD refers to continuous delivery or continuous deployment. Both are related concepts that sometimes are used interchangeably. They are about automating further stages of the pipeline, but sometimes they are used individually to illustrate just how much automation is happening.
Continuous Delivery means a developer's changes to an application code is automatically bug tested & pushed to a repository (like GitHub or Bitbucket), where they can then be deployed to a production environment by the operations team. It prevents the problem of poor visibility and communication between development and business teams. The aim of continuous delivery is to ensure that it takes minimal effort to deploy new application code.

Continuous Deployment refers to automatically releasing developer's changes from the repository to the production environment, where it is usable and accessible to the customers. It addresses the problem of overloading the operations team with manual processes of doing things which slows down application delivery. It builds on the benefits of continuous delivery by automating the next stage in the pipeline.

Business Values/Benefits of CI/CD

CI/CD has numerous advantages to businesses. From cost reduction, controlling and protection. Below are five most essential values and benefits that CI/CD adoption would bring to UdaPeople Products.

I. Reduction of cost with Automation

The absence or minimum intervention of humans in the development processes, the lower the amount of time and money spent. CI/CD Automates lots of things. from Infrastructure Creation and Cleanup, Smokes Tests, Automated Rollbacks, and Auto Deployment to Production.

II. Ensures Code Quality

Good code quality ensures that developers don't spend long hours trying to fix bad code. They move to other projects quickly. This way time and money is saved.

III. Simplified Rollback

A CI/CD pipeline gives developers the ability to fail fast, which helps in recovering even faster. With this, If code is pushed to production and issues arise it can be simply and easily rolled back. This reduces time, energy, and resources consumption and leads to faster fixes to problems.

IV. Faster Release Rate

The faster code is released, the more new code is developed, and then released again. The cycle continues. The business benefit of this is that expensive developer resources would not be sitting idle with a successful CI/CD.

V. Fault and Error Isolation

Catching errors helps reduce cost in the sense that developers spend less time on issues from newly added code.

Before the emergence of CI/CD in software development, development teams would know there was an issue with code, but would struggle to know exactly where the problem was happening. CI/CD and its automated testing has changed that setback. Developers can now easily identify and then isolate code faults, dramatically improving productivity. This saves time and time means money.