Making the Case for CI/CD

Why Adopting CI/CD is in Our Best Interests

What is CI/CD?

CI/CD stands for Continuous Integration, Continuous Delivery and Continuous Deployment. It embodies a set of culture, practices and principles that seeks to help Software development teams build high quality software quickly and efficiently.

CI/CD typically consists of a pipeline of all the stages involves from the beginning of a project to the end. It incorporates a high quality monitoring, testing and automation to software development and deployment.

CI - Continuous Integration

Continuous Integration is a practice in which all developers working on a project commit their changes to a shared repository frequently. Typically developers work on different branches of a project and on different environments. These changes are then merged when they are then merged a tedious, manually process that cause a lot of problems because changes made in isolation can cause conflicts and problems in the working pipeline.

CI ensures that these merges occur frequently and are validated automatically to ensure that that there are no conflicts.

Continuous Delivery and Deployment

Following a successful CI stage, Continuous delivery automates the release of validated code to a repository. Continuous deployment automates releasing an app to production. This means that means that a developer’s change to a cloud application could go live within minutes of writing it (assuming it passes automated testing). This makes it much easier to continuously receive and incorporate user feedback. Taken together, all of these connected CI/CD practices make deployment of an application less risky, whereby it’s easier to release changes to apps in small pieces, rather than all at once.

Why Should we Adopt CI/CD?

Built on the back of the Agile framework, CI/CD ensures that products are not built in isolation and enables development teams to build

Reduction in cost of operations: Because we are testing and monitoring frequently, it reduces chances of reducing resources associated with passing and deploying features that are likely to fail.

Increase in revenue: Because we are iterating frequently, we are able to deliver what users need quickly and and in turn customers are more willing to pay for our services.

Increase in Productivity: Staff are much more productive as they are free from working on manual tasks and tasks that really matter.

Quality and Reliability: Using CICD techniques ensures that the products are built with the highest quality and standards in the shortest possible time and delivered on time. Which is good for business reputation and branding.