"Continuous delivery is the ability to get changes of all types – including new features, configuration changes, bug fixes, and experiments – into production, or into the hands of users, safely and quickly, in a sustainable way."

Jez Humble

**Traditional Delivery Process**

Traditional code development divides the responsibilities into three teams: the development 💻, quality assurance 🧪, and release teams ☑️. The development team focuses on coding 🐱 💻 for passing the code onto the quality assurance team. Using the technical documentation, the quality team conducts user acceptance tests 🧪 on the code from debugging to checking whether the code is following the requirements. After passing the quality tests, the production team determines whether it is possible to stage the updates 👍.

The traditional release cycle can take up to several months to ensure that the code is in production 🏭. The delay ⌛ may result in the code being not up to date 📅 and not fitting the purpose of the customer. If the customer discovered a bug, it has been too long for the development team to recall and identify the problem. Consequently, this makes debugging more challenging after deploying the program. With individuals working on the different code sections, it weakens the communication 🗣️ between the different teams, which makes it challenging to form more coherent and consistent code 🖥️. Therefore, we need an alternative approach to improve the software delivery experiences.

**Continuous Delivery**

Continuous Delivery uses a pipeline to automate the changes to ensure a faster deployment. A case study examines the differences between Yahoo! and Flickr's code management 🧑 💼 culture in 2005. While Yahoo focused on traditional coding approaches. Flickr implemented continuous delivery to produce regular updates each day 📆.

**Shape, polygon

Description automatically generated**

The exchange allowed Yahoo! to realise the benefits of continuous integration, reducing downtime. The minor changes in code are reversible and have less risk than implementing a lot of changes in one update. In each update, there will be fewer bugs for the programmer to go through and the client will experience fewer changes.