Why CI/CD = Business success?

- What exactly is **CI/CD**?

CI stands for continuous integration:

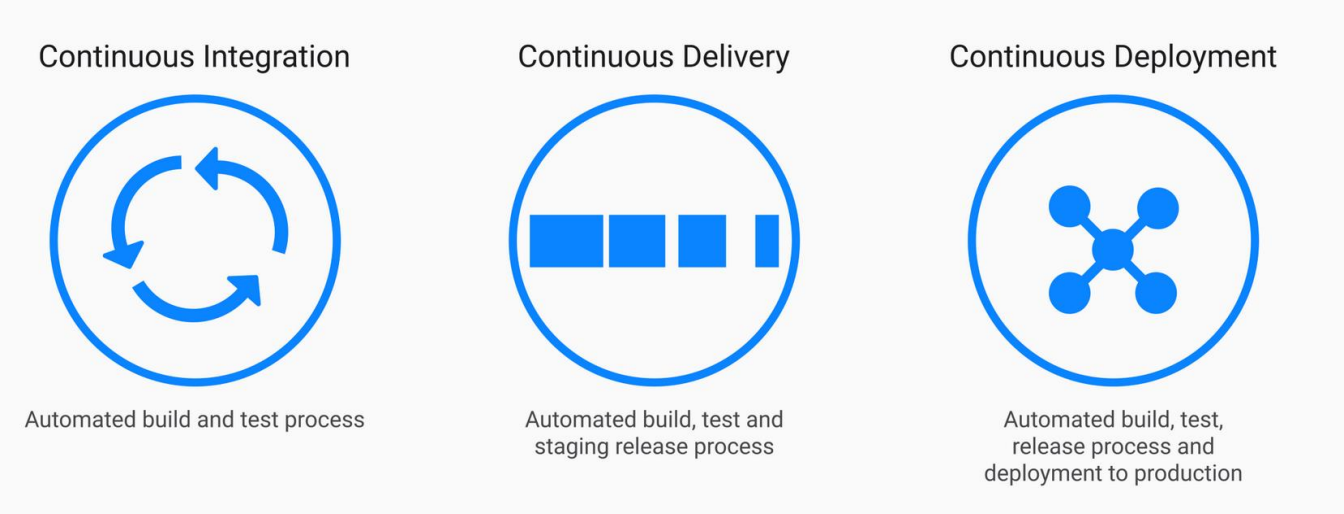
Continuous integration (CI) is practice that involves developers making small changes and checks to their code. Due to the scale of requirements and the number of steps involved, this process is automated to ensure that teams can build, test, and package their applications in a reliable and repeatable way. CI helps streamline code changes, thereby increasing time for developers to make changes and contribute to improved software.

CD stands for continuous deployment:

The final stage of a mature CI/CD pipeline is continuous deployment. As an extension of continuous delivery, which automates the release of a production-ready build to a code repository, continuous deployment automates releasing an app to production. Because there is no manual gate at the stage of the pipeline before production, continuous deployment relies heavily on well-designed test automation.

In practice, continuous deployment 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. There's also a lot of upfront investment, though, since automated tests will need to be written to accommodate a variety of testing and release stages in the CI/CD pipeline.

Graph describing CI/CD in technical terms:



- Almost all of the big successful companies apply **CI/CD**:

  Google, Facebook, LinkedIn, Netflix and many more use CI/CD in their work flows, check the below links for your reference:

  The Facebook Release Process

  Continuous Delivery at Google

  The Software Revolution Behind LinkedIn's Gushing Profits | Business | WIRED

  Deploying the Netflix API

- **CI/CID** Impact on Business:

Cutting Costs:

1. Less bugs in production and less time in testing
2. Prevent embarrassing or costly security holes
3. Less human error, Faster deployments

Reducing Costs:

1. Less developer time on issues from new developer code
2. Less infrastructure costs from unused resources

Increasing Revenue:

1. Less time to market
2. New value-generating features released more quickly

Protecting Revenue:

1. Quick undo to return production to working state in case of failures.
2. Reduced downtime from a deploy-related crash or major bug.