Speed is the key to modern software development. The monolithic all-or-nothing paradigm of traditional waterfall software development has largely been replaced by rapid iterative techniques that support development and release. These techniques go by several names, including continuous integration, continuous delivery and continuous deployment. While each technique offers slight differences, the common emphasis on continuous iteration has changed the nature and power of software development. Businesses can get software to market faster, test innovative new features or architectures while minimizing risk and cost, and effectively refine products over time.

Such iteration relies on well-planned and active pipelines designed to support multiple iterations in various stages of the development cycle simultaneously -- and keep entire development teams constantly busy. As one build is pushed to deployment, the next build undergoes testing, while the very latest build in the cycle is coded.
CI/CD pipelines explained: Everything you need to know.

CI/CD can transform an organization, but there's a lot to consider. This comprehensive guide explains the CI/CD pipeline stages, benefits and challenges, best practices and more.

Continuous integration is a coding philosophy and set of practices that drive development teams to frequently implement small code changes and check them in to a version control repository. Most modern applications require developing code using a variety of platforms and tools, so teams need a consistent mechanism to integrate and validate changes. Continuous integration establishes an automated way to build, package, and test their applications. Having a consistent integration process encourages developers to commit code changes more frequently, which leads to better collaboration and code quality.

Continuous delivery picks up where continuous integration ends, and automates application delivery to selected environments, including production, development, and testing environments. Continuous delivery is an automated way to push code changes to these environments.

## Why CI/CD Matters?

CI/CD practices should matter to you as it helps get continuous feedback not only from your customers but also from your own team. Moreover, in an organization, it can also lead to big advantages. Some of the notable benefits of implementing CI/CD pipelines to your everyday software development process are:

- Reduce costs: Using automation in the CI/CD pipeline helps reduce the number of errors that can take place in the many repetitive steps of CI and CD.
- Smaller code changes: One technical advantage of CI and CD is that it allows you to integrate small pieces of code at one time. This helps developers to recognize a problem before too much work is completed afterward.
- Faster release rate: Failures are detected faster and as such, can be repaired faster, leading to increasing release rates.
- Fault isolations: Designing your system with CI/CD ensures that fault isolations are faster to detect and easier to implement.
- More test reliability: Using CI/CD, test reliability improves due to the bite-site and specific changes introduced to the system, allowing for more accurate positive and negative tests to be conducted.

Continuous Integration

- Detects errors as quickly as possible
  - Fix while fresh in your mind
- Reduces integration problems
  - Smaller problems are easier to digest
  - Don’t compound problems
- Allows teams to develop faster, with more confidence

Continuous Delivery

- Ensures that every change to the system is releasable
- Lowers risk of each release - makes releases “boring”
- Delivers value more frequently
- Get fast feedback on what users care about