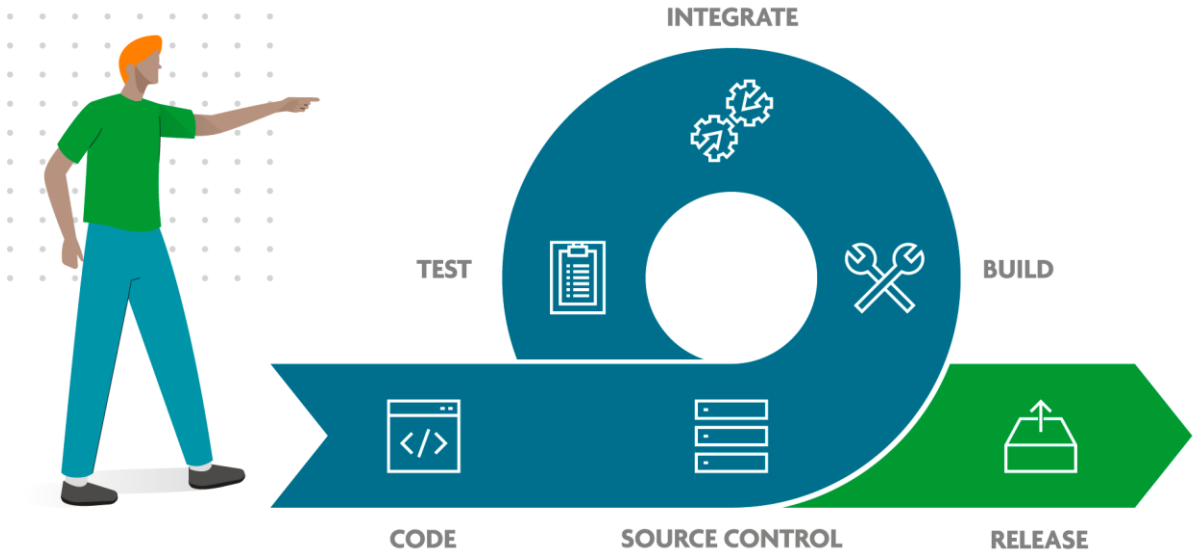
What is CI/CD?

- CI/CD is a method to frequently deliver applications to customers by introducing automation into the stages of app development.
- The main concepts attributed to CI/CD are continuous integration, continuous delivery, and continuous deployment.
- CI/CD is a solution to the problems integrating new code can cause for development and operations teams

Continuous Integration

- Developers practicing continuous integration merge their changes to a main branch as often as possible.
- The developer's changes are validated by creating a build and running automated tests against the build. By doing so, you avoid integration challenges that can happen when waiting for release day to merge changes into the release branch.
- Continuous integration puts a great emphasis on testing automation to check that the application is not broken whenever new commits are integrated into the main branch.



Continuous Delivery

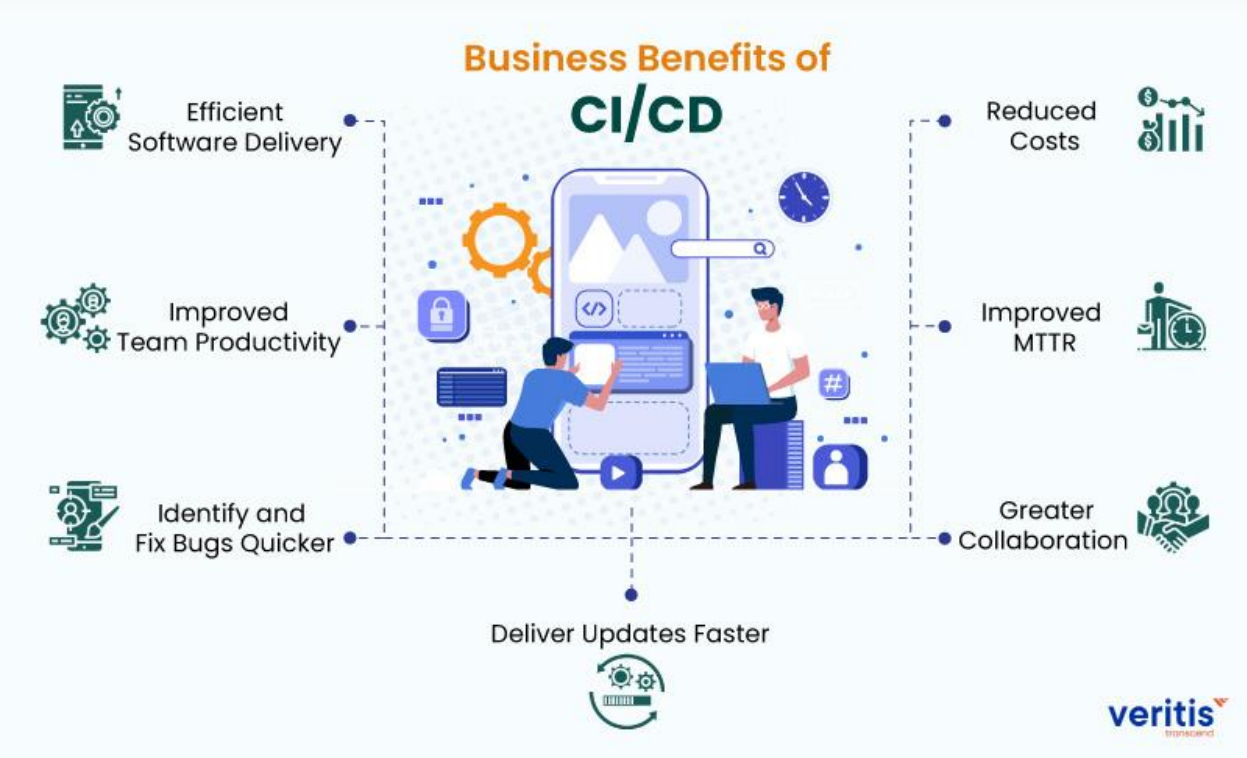
- Continuous delivery is an extension of continuous integration since it automatically deploys all code changes to a testing and/or production environment after the build stage.
- This means that on top of automated testing, you have an automated release process and you can deploy your application any time by clicking a button.

Continuous Deployment

- Continuous deployment goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production.

- Continuous deployment is an excellent way to accelerate the feedback loop with your customers and take pressure off the team as there isn't a "release day" anymore. Developers can focus on building software, and they see their work go live minutes after they've finished working on it.

Benefits of CI/CD



1) Efficient Software Delivery

With continuous delivery, your team can automatically build, test, and prepare code changes for release to production, enabling a more efficient and rapid software delivery.

2) Improved Team Productivity

CI/CD practices improve your team's productivity by freeing developers from manual tasks, reducing the number of errors and bugs deployed to customers.

3) Identify and Fix Bugs Quicker

With CI/CD, your team can perform more frequent and comprehensive testing and quickly identify and address bugs earlier. Continuous delivery enables you efficiently perform additional types of code tests. The discipline of 'more tests more frequently' enables teams to deliver quality code with a high assurance of stability and security.

4) Deliver Updates Faster

Continuous delivery enables faster and frequent delivery of updates to customers. When implemented properly, continuous delivery will help you always have a deployment-ready build artifact that has passed through a standardized test process.

5) Greater Collaboration

CI/CD promotes closer team communication by enabling development, operational, management, and QA teams to collaborate on technologies, practices, and priorities.

6) Improved MTTR

Smaller code changes and quick fault isolation enabled by CI/CD practices reduce the Mean Time To Resolution (MTTR). CI/CD plays a vital role in keeping failures to a bare minimum and quickly recovering from any failures that do happen.

7) Reduced Costs

Automation in the CI/CD pipeline minimizes the number of errors that can occur in the many repetitive steps of CI and CD. It also frees up developers' time that could be spent on product development as there won't be many code changes to fix since the errors are identified quickly.