

CICD Benefits

UdaPeople



What is CI/CD



- CI (Continuous Integration) :-

The practice of merging all developers working copies to a shared mainline several times a day.

- CD (Continuous Delivery) :-

A developer's changes to an application are automatically bug tested and uploaded to a repository (like GitHub or a container registry), where they can then be deployed to a live production environment by the operations team. It's an answer to the problem of poor visibility and communication between dev and business teams. To that end, the purpose of continuous delivery is to ensure that it takes minimal effort to deploy new code.

What is CI/CD

- **CD (Continuous Deployment) :-**

Refer to automatically releasing a developer's changes from the repository to production, where it is usable by customers. It addresses the problem of overloading operations teams with manual processes that slow down app delivery. It builds on the benefits of continuous delivery by automating the next stage in the pipeline.



CI/CD fundamentals



There are eight fundamental elements of CI/CD that help ensure maximum efficiency for your development lifecycle. They span development and deployment. Include these fundamentals in your pipeline to improve your DevOps workflow and software delivery:

1. A single source repository

Source code management (SCM) that houses all necessary files and scripts to create builds is critical. The repository should contain everything needed for the build. This includes source code, database structure, libraries, properties files, and version control. It should also contain test scripts and scripts to build applications.

2. Frequent check-ins to main branch

Integrate code in your trunk, mainline or master branch — i.e., trunk-based development — early and often. Avoid sub-branches and work with the main branch only. Use small segments of code and merge them into the branch as frequently as possible. Don't merge more than one change at a time.

3. Automated builds

Scripts should include everything you need to build from a single command. This includes web server files, database scripts, and application software. The CI processes should automatically package and compile the code into a usable application.

4. Self-testing builds

CI/CD requires continuous testing. Testing scripts should ensure that the failure of a test results in a failed build. Use static pre-build testing scripts to check code for integrity, quality, and security compliance. Only allow code that passes static tests into the build.

CI/CD fundamentals

5. Frequent iterations

Multiple commits to the repository results in fewer places for conflicts to hide. Make small, frequent iterations rather than major changes. By doing this, it's possible to roll changes back easily if there's a problem or conflict.

6. Stable testing environments

Code should be tested in a cloned version of the production environment. You can't test new code in the live production version. Create a cloned environment that's as close as possible to the real environment. Use rigorous testing scripts to detect and identify bugs that slipped through the initial pre-build testing process.

7. Maximum visibility

Every developer should be able to access the latest executables and see any changes made to the repository. Information in the repository should be visible to all. Use version control to manage handoffs so developers know which is the latest version. Maximum visibility means everyone can monitor progress and identify potential concerns.

8. Predictable deployments anytime

Deployments should be so routine and low-risk that the team is comfortable doing them anytime. CI/CD testing and verification processes should be rigorous and reliable, giving the team confidence to deploy updates at any time. Frequent deployments incorporating limited changes also pose lower risks and can be easily rolled back.

Why CI/CD is important ?

- The business impact of CI/CD:-
- **Accelerated time to value** : When you can deploy anytime, you can bring products and new features to **market faster**. Your development **costs are lower**, and a **faster** turnaround frees your team for other work.
- **Hit dates more reliably** : Removing deployment bottlenecks and making deployments predictable can remove a lot of the uncertainty around hitting key dates. Breaking work into smaller, manageable bites means it's easier to complete each **stage on time and track progress**.
- **Less fire fighting** : Testing code more often, in smaller batches, and earlier in the development cycle can seriously cut down on fire drills. This results in a **smoother development cycle** and less team stress.



Why CI/CD is important ?

- The business impact of CI/CD:-
- **Reduce burnout** : Research shows that continuous delivery measurably **reduces deployment pain** and team burnout. Developers experience **less frustration** and strain when working with CI/-CD processes.
- **Less context switching** : Getting real-time feedback on the code developers commit makes it easier to work on one thing at a time and **minimize cognitive load**. By working with small sections of code that are automatically tested, developers can debug code quickly while their minds are still fresh from programming.
- **Recover faster**: CI/CD makes it **easier to fix issues** and **recover from incidents (MTTR)**. Continuous deployment practices mean frequent small software updates so when bugs appear, it's easier to pin them down.



Why CI/CD is important ?

Action	Value
New value-generating features released more quickly	Increase Revenue
Less time to market	Increase Revenue
Less human error, Faster deployments	Cost Avoidance
Quick undo to return production to working state	Protect Revenue



Why CI/CD is important ?

CI/CD automates the process of integrating, releasing, and deploying software while removing traditional roadblocks. It supports the larger goal of agile methodology to accelerate the software development lifecycle, and it supports the DevOps approach of aligning development and operations teams.

- **Ship software quickly and efficiently:** CI/CD pipelines move applications from the coding to deployment phases at scale, ensuring that the pace of development matches the needs of the business.
- **Increase productivity:** By implementing automated processes, development and operations teams are no longer spending time merging, building, testing, releasing, and deploying software manually. Instead, they can focus on writing better code and monitoring deployments for issues.
- **Reduce risk on delivery:** Testing every change before it's deployed ensures that the result will be a higher quality product and lower the rate of bugs in production. Customers will receive a better product, and the development team will spend less time fixing urgent defects discovered after release.
- **Incorporate user feedback faster:** CI/CD removes traditional roadblocks for development and operations teams, enabling the faster release of new features to meet users' needs. This will increase customer satisfaction and provide valuable insights into the capabilities that users value for future projects.
- **Standardize processes:** Automating the merge, test, delivery, and deployment processes means that they will always follow the same structure. This standardizes the pipeline, whereas manual execution of these tasks always comes with the risk of human error, such as executing tests in a different order.

