Continuous Integration Vs Continuous Deployment Vs Continuous Delivery

What is Continuous Integration?

Continuous integration is a software development method where members of the team can integrate their work at least once a day. In this method, every integration is checked by an automated build to search the error.

In continuous integration after a code commit, the software is built and tested immediately. In a large project with many developers, commits are made many times during a day. With each commit code is built and tested. If the test is passed, build is tested for Deployment. If the Deployment is a success, the code is pushed to production. This commit, build, test, and deploy is a continuous process, and hence the name continuous integration/deployment.

What is Continuous Delivery?

Continuous delivery is a software engineering method in which a team develops software products in a short cycle. It ensures that software can be easily released at any time.

The main aim of continuous delivery is to build, test, and release software with good speed and frequency. It helps you to reduce the cost time and risk of delivering changes by allowing for frequent updates in production.

What is Continuous Deployment

Continuous deployment is a software engineering process in which product functionalities are delivered using automatic deployment. It helps testers to validate whether the codebase changes are correct and stable or not.

The team can achieve continuous deployment by relying on infrastructure that automates different testing steps. Once each integration meets this release criteria, the application is updated with a new code.

KEY DIFFERENCES:

- CI is an approach of testing each change to codebase automatically whereas Continuous Delivery is an approach to obtain changes of new features, configuration, and bug fixes. On the other hand, Continuous Deployment is an approach to develop software in a short cycle.

- CI is performed immediately after the developer checks- in. While in Continuous Delivery, developed code is continuously delivered until the programmer considers it is ready to ship and in Continuous Deployment, developers deploy the code directly to the production stage when it is developed.
- CI uses unit tests on the contrary Continuous Delivery uses business logic tests. In Continuous Deployment any testing strategy is used.
- CI refers to the versioning of source code whereas Continuous Delivery refers to the logical evolution of CI and Continuous Deployment refers to automated implementations of the source code.

Difference between CI vs CD vs CD

Here is an important difference between CI vs CD vs CD.

Continuous Integration	Continuous Delivery	Continuous Deployment
CI is an approach of testing each change to codebase automatically.	CD is an approach to obtain changes of new features, configuration, and bug fixes.	CD is an approach to develop software in a short cycle.
CI refers to the versioning of source code.	CD refers to the logical evolution of CI.	CD refers to automated implementations of the source code.
CI focuses on automation testing to determine that the software has no errors or bugs.	Focuses on releasing new changes to your clients properly.	Emphasis on the change in all stages of your production pipeline.
CI is performed immediately after the developer checks-in.	In CD, developed code is continuously delivered until the programmer considers it is ready to ship.	In CD, developers deploy the code directly to the production stage when it is developed.
It helps you to identify and rectify issues early.	It allows developers to check software updates.	It enables you to rapidly deploy and validate new features and ideas.
It uses unit tests.	It uses business logic tests.	Any testing strategy is performed.
Development team sends continuous code merging requests even when the testing process is running.	You deliver code for review that can be batched for release.	Deploy code using an automated process.
You require a continuous integration server to monitor the main repository.	You require a strong foundation in continuous integration.	You need a good testing culture.

Advantages of Continuous Integration

Here are the pros/benefits of continuous integration:

- Helps you to build better quality software
- It enables you to conduct repeatable testing.
- CI allows software developers to work independently on features in parallel.
- It can increase visibility and enable greater communication.
- CI process help to scale up Headcount and delivery output of engineering teams.
- Continuous integration helps you to develop a potentially shippable product for a fully automated build.
- Helps you to redue risks by making deployment faster and more predictable
- immediate feedback when an issue arrives.
- Avoid last-minute confusion at the release date, and timing automates the build.
- It reduces risks and makes the deployment process more predictable.
- CI provides instant feedback when there is an issue.
- You can see the integration process in real time.
- It can avoid last-minute hassle at release dates.
- The current build is available constantly.
- Provides shippable products on a regular base.
- It is relatively easy to find a history of the software build.
- CI offers code stability.

Advantages of Continuous Delivery

Here are the pros/benefits of continuous delivery:

- Automate the software release process for making delivery more efficient, rapid, and secure.
- CD practices increase productivity by freeing developers from manual work and complex dependencies.
- It helps you to discover software bugs early in the delivery process.
- CD helps your business team to deliver updates to clients immediately and frequently.
- It ensures the software is always ready to go to production.
- You can release software more frequently, which helps you to get fast feedback from your clients.
- There is less pressure on decisions for small changes.

Advantages of Continuous Deployment

Here are the pros/benefits of continuous Deployment:

- It helps you to automate the repetitive tasks.
- CD makes your deployment flawless without compromising security.

- Easily scale from a single software application to an enterprise IT portfolio.
- You can ship cloud-native as well as traditional applications.
- It gives a single view across all environments and applications.
- You can connect your existing Devops tools and scripts into a proper workflow.
- CD enables you to increase overall productivity.
- You can integrate processes and teams with a unified pipeline.

Disadvantages of Continuous Integration

Here are the cons/ disadvantages of continuous integration:

- Initial setup time and training is required to get acquainted with CI server
- Well-developed test-suite required many resources for CI server.
- It requires additional servers and environments.
- You need a conversion of familiar processes in one project.
- It goes for waiting when multiple developers integrate their code around the same time.
- Your team should write automated tests for each and every new feature or bug fix.
- You require a CI server that monitors the main repository and run the tests for new code commits.
- Developers should merge their changes as more often as possible.
- The unit testing procedure should pass for the Deployment.

Disadvantages of Continuous Delivery

Here are the cons/disadvantages of continuous delivery:

- You should know continuous integration practices before you go for continuous delivery.
- Deployments are still manual, and hence it takes lots of time to deliver the software product.
- The automated tests should be written and function properly.
- Faulty tests can lead to damage while quality testing.
- It requires team coordination because code changes should be collected regularly in an efficient way.
- Continuous delivery requires a reliable and strong integration server for the automation test that is costly.

Disadvantages of Continuous Deployment

Here are the cons/disadvantages of continuous Deployment:

- Your testing culture should be good as the quality of the suite determines how good software releases are.

- Documentation procedures need to keep up with deployment pace.
- Releasing significant changes needs assurance by marketing, help, and support, and other departments.

Continuous Integration Best Practices

Here are some important best practices while implementing Continuous Integration.

- Automate your software build.
- Keep the build as fast as possible.
- Every commit should result in a build
- Automate Deployment
- Commit early and often.
- You should never commit broken code
- Fix build failures immediately.
- Build-in every target environment Create artifacts from every build
- The build of the software needs to be carried out in a manner so that it can be automated
- Do not depend on an IDE
- Build and test everything when it changes
- The database schema counts as everything
- Helps you to find out key metrics and track them visually
- Check-in often and early.
- Stronger source code control.
- Continuous integration is running unit tests whenever you commit code.
- Automate the build and test everyone.
- Keep the build fast with automated Deployment.

Continuous Delivery Best Practices

Here are some important best practices while implementing continuous delivery:

- The first stage must be triggered upon every check-in.
- Each stage should trigger the next one quickly upon successful completion.
- Maintain the version of the source code.
- Perform automated build and Deployment.
- Deploy to one instance of a virtual machine at a time.
- Perform unit and integration tests.
- You have to build your library only once.
- The team should use the same automated release method for each and every environment.
- This method enables you to eliminate conflicts and last-minute problems.
- In case any state fails, you should automatically pause the process and fix the issues.

Continuous Deployment Best Practices

Here are some important best practices while implementing continuous Deployment:

- You should use an issue tracker for the development task.
- In your version controlling system, you should create a branch that contains the issue number and description of any change you have made.
- When software is ready for the Deployment, you can create a pull request for the branch.
- Deployment software to pre-production staging servers.
- Promote your software once you are happy with its quality.

Challenges of Continuous Integration

Here are the challenges of continuous integration:

- It makes the developing process slow.
- Exposes problems and sharing of issues.
- It may lead to a lack of maintenance of version control.
- It can force you to deal with problems.
- Difficulty in building automated code repository.
- Untested or broken code must not be committed.

Challenges of Continuous Delivery

Here are the challenges of continuous delivery:

- You need to keep the continuous delivery efficient without bothering the time.
- You need to cope up with tight deadlines release plan.
- Poor product-specific communication of teams may lead to revisions as well as deployment delays.
- The business team should have the budget to have the infrastructure needed to build more impressive software.
- Monitoring data/ information should be utilized by the research and development team.
- The organization should ensure that how open source software fits into the current workflow.

Challenges of Continuous Deployment

Here are the challenges of continuous deployment:

- CD requires continuous planning to achieve frequent and fast releases.

- Ensure the alignment between the requirement of the business context and application development.
- Rapid delivery must not be isolated to the software development process alone.
- The flow should go with the overall software development cycle.
- Experimental results must be continuously linked with the software roadmap.