CI/CD proposal

Developing and releasing software can be a complicated process, especially as applications, teams, and deployment infrastructure grow in complexity. Often, challenges become more pronounced as projects grow. To develop, test, and release software in a quick and consistent way, developers and organizations have created three related but distinct strategies to manage and automate these processes.

Continuous integration focuses on integrating work from individual developers into the main repository multiple times a day to catch integration bugs early and accelerate collaborative development.

Continuous delivery is concerned with reducing friction in the deployment or release process, automating the steps required to deploy a build so that code can be released safely at any time. Continuous deployment takes this one step further by automatically deploying each time a code change is made.

In another meaning

Continuous integration:

Continuous integration can save a lot of time that the developer takes in integrating and testing the code into the main branch by making it itself.

In fact, Continuous integration encourages developers to integrate their code into the main branch more early and often.

That can minimize the cost of integration a lot by making it an early consideration

As the developers can discover the conflicts between the old and the new code at an early stage so it will take short time if compared to the same issue if discovered at a complex stage.

The idea of continuous integration is that will take the code you have written and just pass it to tests to make sure everything is okay and the code can be deployed safely.

But if everything is not okay it will alert the developing team in order to handle or fix the code to make it run successfully.

It will be better than discovered in the deployment stage that can cost a lot.

Continuous delivery:

Continuous delivery continues what continuous integration begins as if continuous integration finishes testing successfully it will take the code to the next stage as it focuses on automating the whole software delivery process.

It is a mixing of technical and organizational systems.

On the technical side, the deployment leans heavily on deployment pipelines to automate the testing and deployment processes.

A deployment pipeline is an automated system that runs all the test scripts that test all the functionality of the code to make sure every part of the code is run as

it should be, so at any test stage if the code fails it will send an alert to the team or success so it will move to the next stage until it gets deployed.

So the Deployment is automated, so the team will not be worried to build and publish new functionality as it will be deployed as soon as it is pushed to the repo.

On the organization side, the organizational aspects of continuous delivery encourage prioritization of “deployability” as a principle concern. This has an impact on the way that features are built and hooked into the rest of the codebase. Thought must be put into the design of the code so that features can be safely deployed to production at any time, even when incomplete. A number of techniques have emerged to assist in this area.

So in other words

With the presence of CI/CD, it will be easier for the team to be concerned about their field and not to worry about the integration and deployment.

So it will save the cost of the time that the CI/CD do for the developer and it also chooses the best strategy for the deployment so it will make it easier for the developing team and the organization.