CI:

Continuous Integration takes care of integrating the work done by individual developers into a repository. Following the DevOps approach, this is done several times in a day to detect the bugs in the initial phases itself. It increases collaboration among development teams. Generally, the application which is working in the local environment stops working in the production. CI/CD helps in avoiding these sudden shocks by enabling continuous testing along with the development. It involves the automation of each step of a release process. Continuous Deployment enables teams to automatically deploy the changes to the production without any manual intervention.

CD:

After continuous delivery, the software build can be automatically deployed to the production if it passes all the tests. There is no manual intervention, and no one is required to decide when the code is ready to get into production and what needs to be sent to the production. The last step in the CI/CD pipeline will automatically publish the changes to the production. It can save a lot of time, money and effort of the Dev and Ops teams.

There are many CI/CD tools like Jenkins, available in the market that can integrate with the developer tools like Github and Bitbucket. But using these tools you need to manage different applications for each use case. With PWSLab you need not use different tools to achieve different results. It is a single secured application addressing all your needs related to SDLC.

Business Benefits of CI/CD

Implementing continuous integration, developers do frequent code commits which makes it easier to identify defects and quality related issues. On the other hand, following the conventional methods force developers to do the same task again and again. This ultimately affects their efficiency and brings less return on investment. Continuous integration leads to lesser merge conflicts due to the shorter commit cycles which ultimately save a lot of business hours which can be spent on creating more new interesting features.
The first step towards continuous integration is configuring the version control system.
Version-controlling using branches is one such technique. The branching strategy set rules over how new code should be handled.
Developers can trigger builds anytime using the CI/CD. Code commits can trigger the builds in the version control system or on a defined schedule. Teams can set there build schedules according to their team size and deployment frequency. Planning ahead is the best practice to avoid any delays in the deployments and releasing new features.