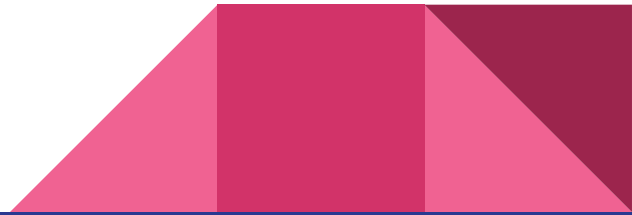
CI/CD

Save money & time with automation super powers!

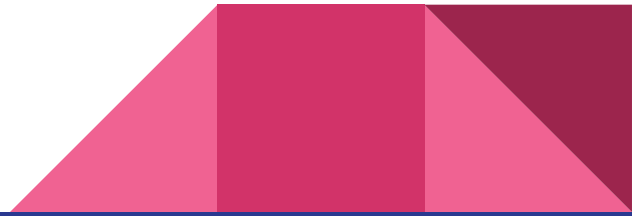
What is CI/CD ?

CI/CD stand for continuous integration and continuous delivery. allows organizations to ship software quickly and efficiently. CI/CD facilitates an effective process for getting products to market faster than ever before, continuously delivering code into production, and ensuring an ongoing flow of new features and bug fixes via the most efficient delivery method.



Continuous Integration (CI)

Continuous integration means that developers frequently merge their code changes to a shared repository. It’s an automated process that allows multiple developers to contribute software components to the same project without integration conflicts. CI involves automated testing whenever a software change is integrated into the repository.



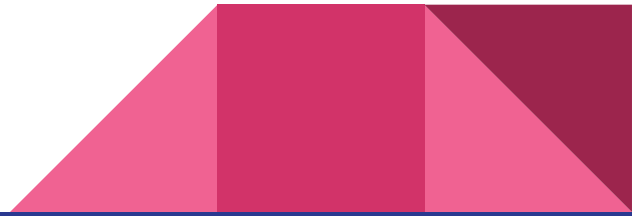
What is CD?

CD can stand for either continuous delivery or continuous deployment. Both involve taking the code continuously integrated and getting it able to deploy to an environment either QA or production. Continuous deployment takes the process one step further and performs the actual deployment to an environment.

CI/CD Benefits

● Make Revenue – Faster and More Frequent Production Deployments ensures more quicker releases. Removal of manual checks before deployment means less time to market.

● Protect Revenue – Automated smoke test reduces downtime due to deploy related crash or a major bug. Automated rollback due to a job failure mea ns a fast undo from production to working state.



CI/CD Benefits (Cont.)

● Avoid Cost – Automation of infrastructure creation hence faster deployment and less human error. Catch unit test failure ensures less bugs in production environment and less time testing.

● Reduce Cost – Automation of infrastructure cleanup prevents unwanted cost on unused resources. Catching compile errors after merging reduces time spent on issues from new developer code.

● Avoid Cost – Automation of infrastructure creation hence faster deployment and less human error. Catch unit test failure ensures less bugs in production environment and less time testing. Detecting security vulnerabilities avoids future embarrassment from security attacks.

● Reduce Cost – Automation of infrastructure cleanup prevents unwanted cost on unused resources. Catching compile errors after merging reduces time spent on issues from new developer code.