CI/CD Pipelines

Continuous integration

- As code is changed, it is automatically and seamlessly merged with the central repository, or “mainline.” This is done frequently to eliminate the possibility that some or all of the development team is working with out-of-date copies of the code, which could cause an application to break.

Continuous delivery

- When checked-in mainline code is ready for deployment to end-users, it is then alpha and beta tested in “staging,” or non-production environments, and then manually released to production.

Continuous deployment

- While with continuous delivery, customers do not receive the update until it is manually released, continuous deployment extends the use of automation in the testing environment and automatically releases the newest changes to production as soon as they have passed all required tests.

DevOps

- This term is ubiquitous in the CI/CD world and is often treated as if the two were identical. Dev/Ops refers to the underlying mindset more than to a particular methodology. It refers to the cultural change required to foster a faster release cycle, create collaborative interdisciplinary teams (comprising developers, operations, quality assurance, and management), and introduce automation into the process.

Business Benefits of CI/CD Pipelines

- **Better quality software**. With CI/CD, your developers are handling smaller sections of code, meaning that if problems do emerge, it will be simpler and quicker to find a resolution. It also lowers the chance that errors make it into the production environment. Additionally, by incorporating automated testing, CI/CD can resolve many bugs before they make it into end users' hands, boosting user satisfaction.

Maximum consumer demand satisfaction

- CI/CD helps you adopt and incorporate a customer-first approach. When you release a product, it carefully monitors the initial actions of the customers and maintains a record of the results. As a result, you can study the kind of impression your product creates on the customers.

Reduce Costs

- Automation in the CI/CD pipeline reduces the number of errors that can take place in the many repetitive steps of CI and CD. Doing so also frees up developer time that could be spent on product development as there aren’t as many code changes to fix down the road if the error is caught quickly. Another thing to keep in mind: increasing code quality with automation also increases your ROI.

CI\CD revenue/cost-related terms

CI/CD Language	Captured Value	Translation
Catch Compile Errors After Merge	Reduce Cost	Less developer time on issues from new developer code
Catch Unit Test Failures	Avoid Cost	Less bugs in production and less time in testing
Detect Security Vulnerabilities	Avoid Cost	Prevent embarrassing or costly security holes
Automate Infrastructure Creation	Avoid Cost	Less human error, Faster deployments
Automate Infrastructure Cleanup	Reduce Cost	Less infrastructure costs from unused resources
Faster and More Frequent Production Deployments	Increase Revenue	New value-generating features released more quickly
Deploy to Production Without Manual Checks	Increase Revenue	Less time to market
Automated Smoke Tests	Protect Revenue	Reduced downtime from a deploy-related crash or major bug
Automated Rollback Triggered by Job Failure	Protect Revenue	Quick undo to return production to working state

Conclusion

- There are many tools that can help enable a smoother transition to a CI/CD process. Testing is a large part of that process because even if you are able to make your integrations and delivery faster, it would mean nothing if was done so without quality in mind. Also, the more steps of the CI/CD pipeline that can be automated, the faster quality releases can be accomplished.