CI/CD INTRODUCTION

1. Concepts
2. Benefits
3. Conclusion

CONCEPTS - CI

- Continuous integration (CI) is the practice of automating the integration of code changes from multiple contributors into a single software project. It's a primary DevOps best practice, allowing developers to frequently merge code changes into a central repository where builds and tests then run. Automated tools are used to assert the new code's correctness before integration.
- A source code version control system is the crux of the CI process. The version control system is also supplemented with other checks like automated code quality tests, syntax style review tools, and more.

CONCEPTS - CD

- Continuous deployment (CD) is a software release process that uses automated testing to validate if changes to a codebase are correct and stable for immediate autonomous deployment to a production environment.
- The software release cycle has evolved over time. The legacy process of moving code from one machine to another and checking if it works as expected used to be an error-prone and resource-heavy process. Now, tools can automate this entire deployment process, which allows engineering organizations to focus on core business needs instead of infrastructure overhead.

BENEFITS – FASTER TIME TO MARKET

- The primary goal of a CI/CD pipeline is to deliver working software to users quickly and frequently.
- Reduced risk: having a shorter time to market doesn't just help you keep up with the competition. Rapid releases provide an opportunity for product managers and marketing professionals to engage more closely with the development process.
- Shorter review time: with continuous integration, developers are encouraged to commit their code changes more frequently – at least once a day as a rule of thumb. Sharing code with the rest of the team regularly not only ensures everyone is building on the same foundation, but also results in faster code reviews and makes it easier to integrate changes.

BENEFITS – BETTER CODE QUALITY

- Testing your code’s behavior is an essential step in the software release process but doing it thoroughly can also be extremely time consuming.
- Smoother path to production: as we all know, practice makes perfect, and what’s true of shooting hoops or mastering scales also applies to software releases.
- Faster bug fixes: even with improved code quality thanks to automated testing, bugs will still occasionally sneak their way through to production.
- Efficient infrastructure: automation is a central part of any CI/CD pipeline, serving to make the release process repeatable and reliable.
- Measurable progress: many of the tools available to support automated CI/CD also instrument the process, providing you with a whole host of metrics from build times to test coverage, defect rates to test fix times.

BENEFITS – MAXIMIZED CREATIVITY

- As we’ve seen, building a CI/CD pipeline eliminates waste and helps create a leaner, more efficient software development and release process.
- By using computers to perform repetitive tasks, an automated process also frees up individuals to be creative. Instead of following manual test scripts, refreshing environments, or deploying updates, you can focus on solving problems and experimenting with solutions.
- Having the scope to be more creative and add value in what you do leads to improved job satisfaction, which encourages people to contribute more, attracts more talent and improves staff retention. In turn that benefits your organization, your product, your users, and ultimately your bottom line.

CONCLUSION

- Reduce costly downtime: Automation improves reliability of deployments, reduces impact on revenue or credibility that could potentially result from extensive downtime.
- Minimize troubleshooting: Product development and release processes are more reliable when automated. Developers waste less time bug-fixing and can focus on improving quality, increasing DevOps ROI and pushing out new features faster.
- Lower learning curve: Automated CI/CD pipelines free up developers from time consuming manual tasks. The learning curve and training costs for new team members are dramatically reduced.
- Bring down staff costs: All successful software products and services evolve significantly in their lifetimes. Automation eliminates many fixed costs embedded in the release process, substantially reducing the cost of making incremental changes to software as they evolve.