CI/CD Fundamentals and Benefits

By:Abdalla Amin

Fundamentals of CI/CD

- **Continuous Integration:** The practice of merging all developers' working copies to a shared mainline several times a day.
- **Continuous Delivery:** An engineering practice in which teams produce and release value in short cycles.
- **Continuous Deployment:** A software engineering approach in which the value is delivered frequently through automated deployments.

Fundamentals of CI/CD

- Pipeline: A set of data processing elements connected in series, where the output of one element is the input of the next one.
- Infrastructure as Code: The management of infrastructure using code.
- uProvisioning: The process of setting up IT infrastructure.
- Artifact: A product of some process applied to the code repository.
- DevOps: A set of practices that works to automate and integrate the processes between software development and IT teams.
- Testing: A practice that seeks to ensure the quality of the software.

Benefits of CI/CD

- Less developer time on issues from new developer code
- Less bugs in production and less time in testing
- Prevent embarrassing or costly security holes
- Less human error, Faster deployments
- Less infrastructure costs from unused resources
- New value-generating features released more quickly
- Less time to market
- Reduced downtime from a deploy-related crash or major bug
- Quick undo to return production to working state

Best Practices for CI/CD

- Fail Fast: Set up your CI/CD pipeline to find and reveal failures as fast as possible. The faster you can bring your code failures to light, the faster you can fix them.
- Measure Quality: Measure your code quality so that you can see the positive
- effects of your improvement work (or the negative effects of technical debt).
- Only Road to Production: Once CI/CD is deploying to production on your behalf, it must be the only way to deploy. Any other person or process that
- meddles with production after CI/CD is running will inevitably cause CI/CD to become inconsistent and fail.
- Maximum Automation: If it can be automated, automate it. This will only improve your process!
- Config in Code: All configuration code must be in code and versioned
- alongside your production code. This includes the CI/CD configuration files!

Value FrameWork Focus

- Increase Revenue
- Protect Revenue
- Reduce Cost
- Avoid Cost

Apply Value FrameWork on CI/CD

Catch Compile Errors After Merge:

- **Technical:** CI/CD is able to catch compile errors after merge
- **Translation:** This ability could help us reducing coasts, because there will be less developer time on issue from new developer code

- **Catch Unit Test Failures:**
- **Technical:** CI/CD is able to catch unit test failures
- **Translation:** This ability could help us avoiding coasts, because there will be less bugs on production and less time on testing

- **Detect Security Vulnerabilities:**
- Technical: CI/CD is able to detect security vulnerabilities
- Translation: This ability could help us avoiding coasts, because we can prevent embarrassing or costly security holes

- **Automate Infrastructure Creation:**
- Technical: CI/CD is able to automate infrastructure creation
- Translation: This ability could help us avoiding coasts, because there will
- be less human error, Faster deployment

- **Automate Infrastructure Cleanup:**
- Technical: CI/CD is able to automate infrastructure cleanup
- Translation: This ability could help us reducing coasts, because there will be Less infrastructure costs from unused resources
- **Faster and More Frequent Production Deployments:**
- Technical: CI/CD is able to have faster and more frequent production deployments
- Translation: This ability could help us increasing revenue, because there will be a new value-generating features released more quickly