Fundamentals and Benefits of CI/CD to Achieve, Build, and Deploy Automation for UdaPeople

WHAT IS CI/CD?

CI means Continuous Integration, which is a practice that ensures high code quality by running automated builds, scans, and tests on code that is pushed to a repository. This ensures bugs are caught faster and vulnerabilities get fixed as early as possible before the code gets deployed.

CD means Continuous Delivery or Continuous Deployment. This basically ensures code changes are automatically prepared for a release to production. This results in a smooth release of new features in a reliable and automated way, leading to faster time to market.

Benefits of Continuous Integration and Deployment

- Reduce cost by catching compile errors, which enables developers spend less time on issues from new developer code.
- Avoid unnecessary costs by catching unit test failures early. These leads to fewer bugs in production and less time in testing.
- Detecting security vulnerabilities avoids costs and prevents embarrassing or costly security holes.
- Automating Infrastructure creation and configuration prevents the risk of human error and faster deployments.
- Automating Infrastructure cleanup reduces cost and less infrastructure costs from unused resources.
- Faster and more frequent deployments which would lead to increase in revenue.
- New features released more quickly which generates revenue.
- Deploy to production using automated checks increases revenue and enables features to ship with less time to market.

- Automated smoke tests protect revenue by ensuring the application returns expected response. This reduces downtime from a deploy-related crash or major bug.
- Automated rollback triggered by job failure protects revenue and enables the ability to return production to working state when there is a major bug that makes the application unusable.

STEPS IN CI/CD PROCESS

A typical CI process would start with a change made in the code and pushed to a branch. The build could be triggered on pull requests or just push depending on the use case. Tests and scans are then carried out. Based on the build, tests and scans results, the pipeline could stop at this point and prevent merge of the code. If the code is good and ready for release, then the rest of the continuous delivery process would run. Automating creation and configuration of the infrastructure, deployment of application and smoke tests would