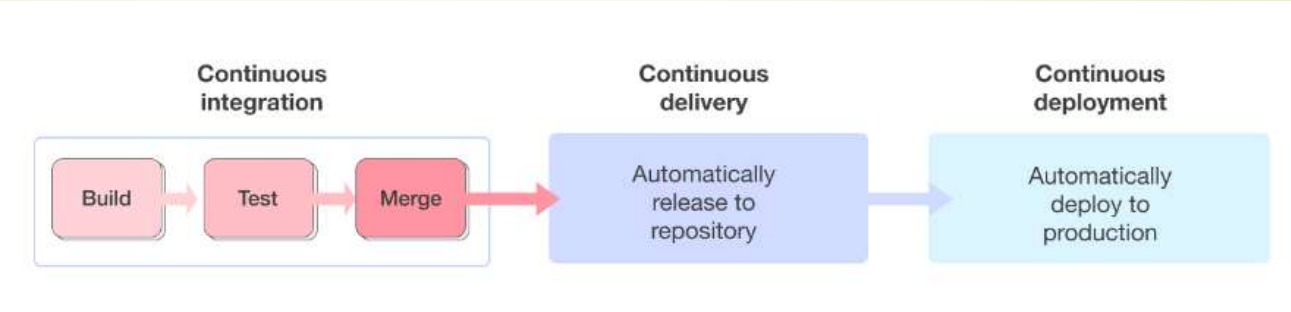
CI/CD

Continuous Integration/Continuous Deployment

Give Your Application Auto-Deploy Superpowers

NOURHAN SABER

Understanding the key difference between CI & CD



Continuous integration

Continuous integration enables merging code changes back to a shared branch, commonly referred to as the trunk, multiple times daily. After the changes are merged, they are validated by automatic application building and testing, including unit and integration tests, that ensure the changes do not break the application. Even if the automated tests find conflicts between the two codes, CI simplifies fixing them.

Continuous delivery

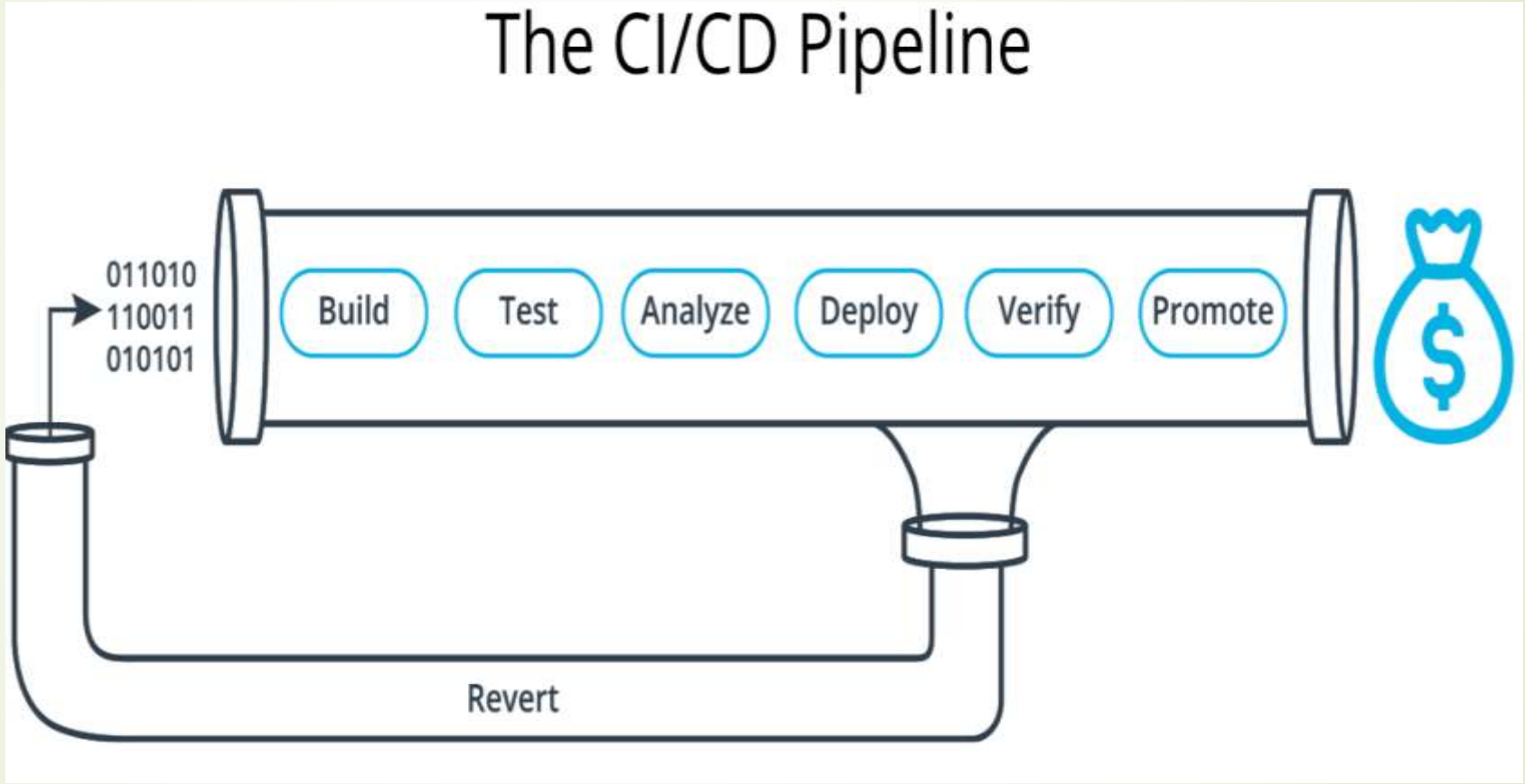
After automating the builds and testing in CI, the role of continuous delivery is to automate the release of the validated code to a repository. Therefore, an effective continuous delivery process requires a built-in CI in the development pipeline because the entire purpose of continuous delivery is to maintain a codebase that is always ready to be deployed in a production environment.

Continuous deployment

The concluding stage of a CI/CD pipeline is continuous deployment, which acts as an extension of continuous delivery. It is responsible for automating the release of a build ready for production. Moreover, due to the lack of manual intervention, well-structured test automation is the foundation of continuous deployment.

It facilitates a change to go live in an application in a few minutes of writing it. The entire practice of deploying applications becomes less risky, and changes are released in small pieces instead of as a whole.

Workflow


The CI/CD Pipeline
011010
110011
010101
Build
Test
Analyze
Deploy
Verify
Promote
$
Revert

Benefits of CI/CD

Higher efficiency

Increased productivity is one of the leading advantages of a CI/CD pipeline. You should automate your process if you have a review process that includes deploying code to development, testing, and production environments and entering multiple commands across several domains. This creates the need for a CI/CD framework.

Faster product delivery

With a smooth CI/CD workflow, multiple daily releases can become a reality. Teams can automatically build, test, and deliver features with minimal manual intervention.

Log generation

Observability is pivotal for DevOps. If something isn't right, you need to figure out why. You'll need a way to track the system's performance over time to determine essential performance indicators. Observability is a technical tool that aids in this endeavor.

Cost-effectiveness

The CI/CD pipeline takes a different approach to software delivery. It can be compared to an assembling unit's delivery pipeline. In any business situation, time and assets are essential. Firms are expected to respond to client demands quickly and effectively with such requirements.

Reduce Cost:

Catch Compile Errors After Merge : Less developer time on issues from new developer code.
Automate Infrastructure Cleanup: Less infrastructure costs from unused resources.

Avoid Cost:

Catch Unit Test Failures: Less bugs in production and less time in testing.
Detect Security Vulnerabilities: Prevent embarrassing or costly security holes.
Automate Infrastructure Creation: Less human error, Faster deployments.

Increase Revenue:

Faster and More Frequent Production Deployments: New value-generating features released more quickly.
Deploy to Production Without Manual Checks: Less time to market.

Protect Revenue:

Automated Smoke Tests: Reduced downtime from a deploy-related crash or major bug.
Automated Rollback Triggered by Job Failure: Quick undo to return production to working state.