CI/CD

A better way to build and ship products to market

Continuous Integration (CI)

01

REDUCE WORKING TIME WITH ALL DEVELOPERS DUE TO FASTER IMPLEMENTATION.

02

HIGH QUALITY DEPLOYABLE ARTIFACT.

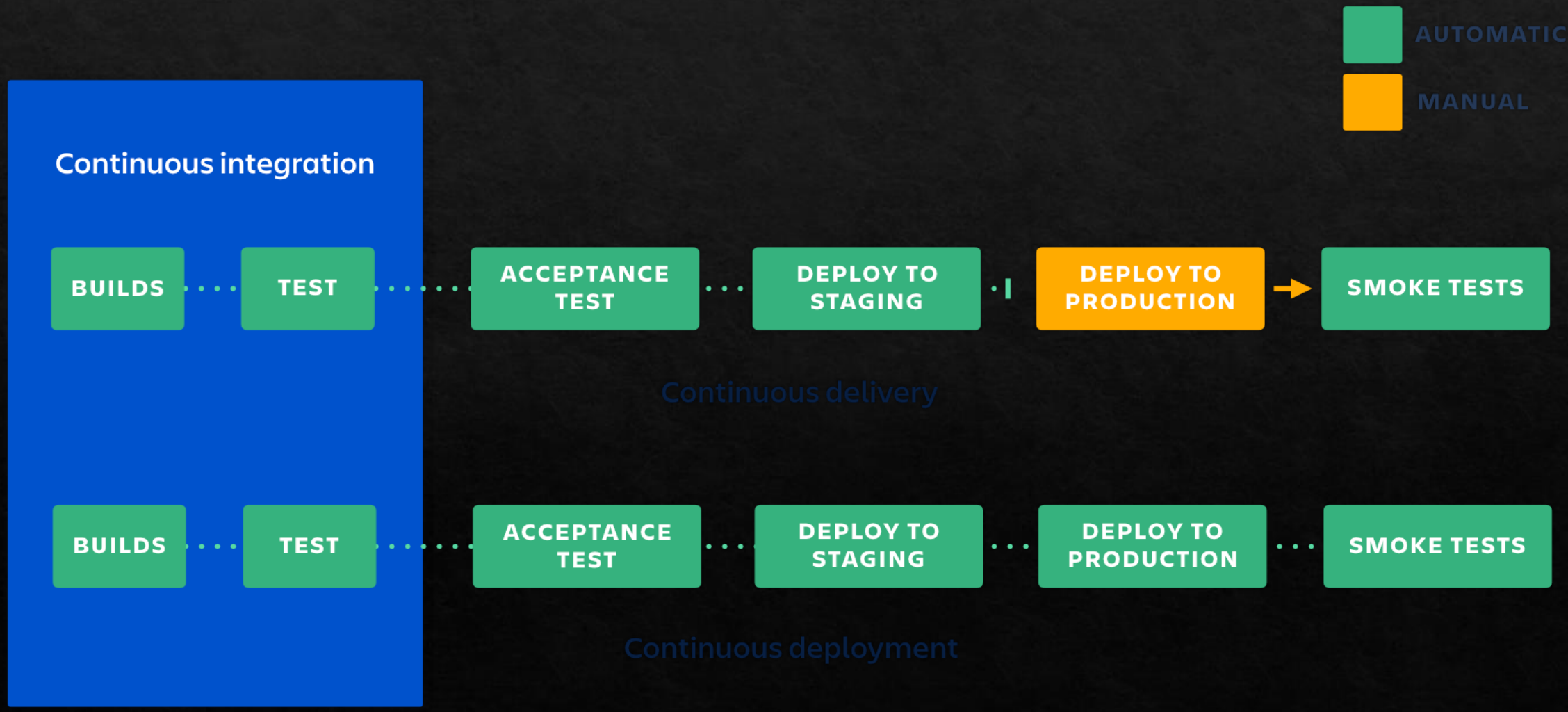
03

IT'S ALL ABOUT DEV AND WORKFLOW.

Industry-Validated Best Practices for Implementing CI/CD

- The *Plan* phase often combines practices from Scrum and Agile to enable frequent micro incremental releases.
- The *Code* phase focuses on core development tasks from within IDEs and appropriate sandboxing and frameworks.
- The *Build* phase rapidly and incrementally merges code commits with some testing and security validation.
- The *Test* phase focuses on automated verification of enhancements, often incorporating test-driven deployment practices. Testing is sometimes incorporated as part of the Build phase, and generally extends in some way to all phases of the CI/CD process to ensure continuous feedback and improvement.

Continuous Deployment



Why CI/CD ?

- Continuous deployment is a strategy in software development where code changes to an application are released automatically into the production environment. This automation is driven by a series of predefined tests. Once new updates pass those tests, the system pushes the updates directly to the software's users.
- Continuous deployment offers several benefits for enterprises looking to scale their applications and IT portfolio. First, it speeds time to market by eliminating the lag between coding and customer value—typically days, weeks, or even months.
- In order to achieve this, regression tests must be automated, thereby eliminating expensive manual regression testing. The systems that organizations put in place to manage large bundles of production change—including release planning and approval meetings—can also be eliminated for most changes.