Build and Deployment Automation

- **Build automation** is the process of automating the creation of a software build and the associated processes including: compiling computer source code into binary code, packaging binary code, and running automated tests.
- **Deployment automation** is what enables you to deploy your software to testing and production environments with the push of a button. Automation is essential to reduce the risk of production deployments. It's also essential for providing fast feedback on the quality of your software by allowing teams to do comprehensive testing as soon as possible after changes.

An automated deployment process has the following inputs:

1. Packages created by the continuous integration (CI) process (these packages should be deployable to any environment, including production).
2. Scripts to configure the environment, deploy the packages, and perform a deployment test (sometimes known as a *smoke test*).
3. Environment-specific configuration information.

- A deployment pipeline is an automated manifestation of your process for getting software from version control into the hands of your users.
- That process involves building the software, followed by the progress of these builds through multiple stages of testing and deployment.
- This, in turn, requires collaboration between many individuals, and perhaps several teams.
- The deployment pipeline models this process, and its incarnation in a continuous integration and release management tool is what allows you to see and control the progress of each change as it moves from version control through various sets of tests and deployments to release to users.

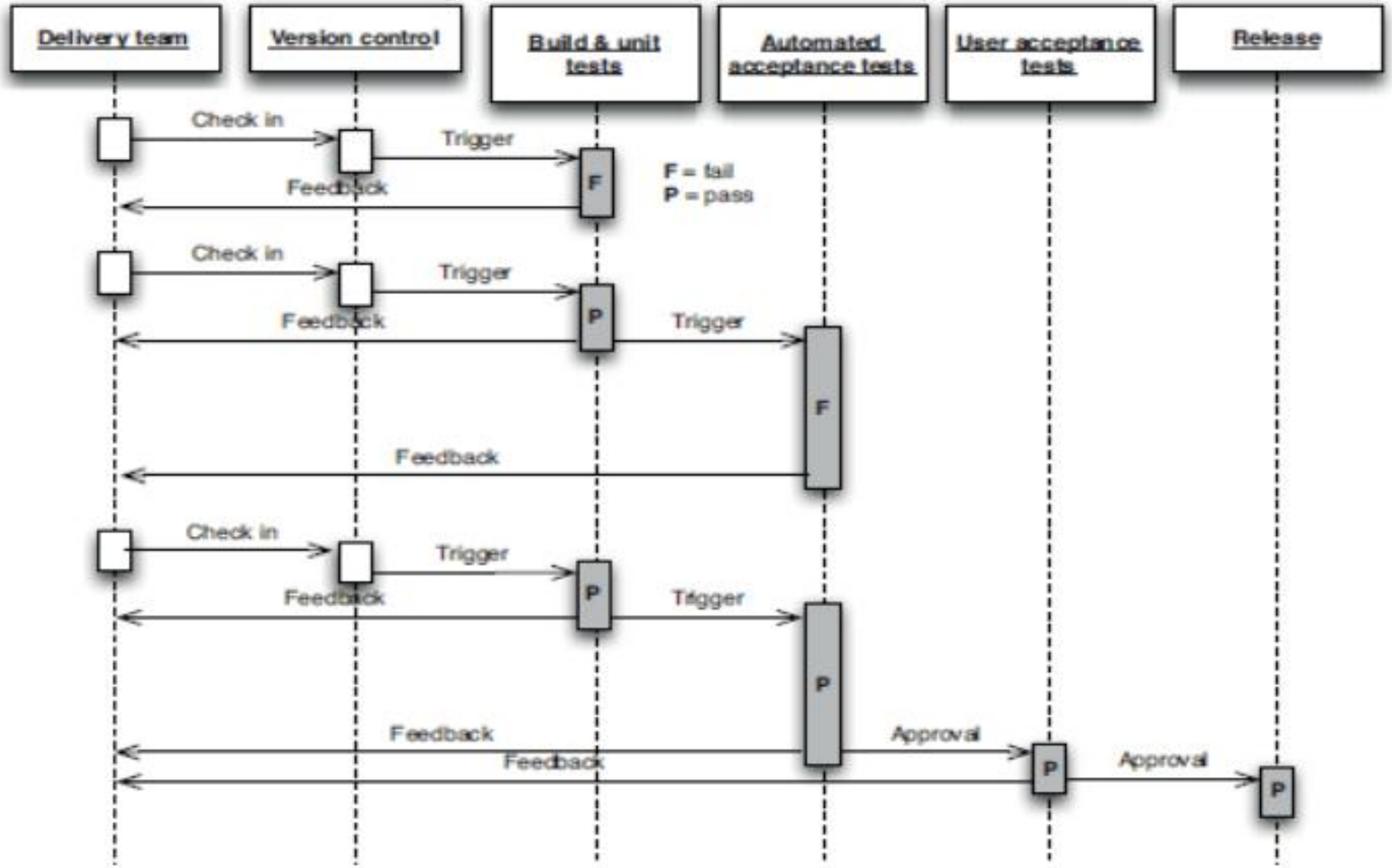


Figure 5.2 *Changes moving through the deployment pipeline*

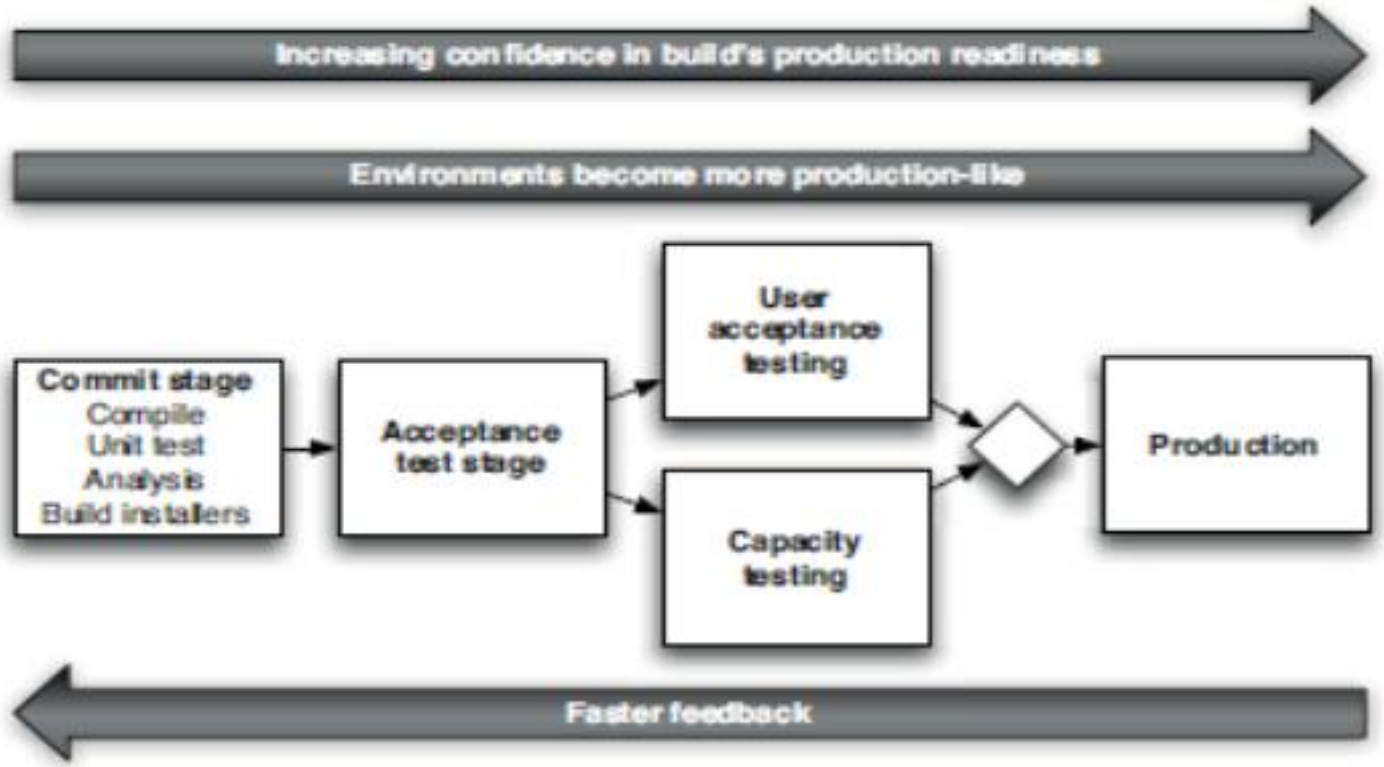


Figure 5.3 *Trade-offs in the deployment pipeline*

- *The commit stage* asserts that the system works at the technical level. It compiles, passes a suite of (primarily unit-level) automated tests, and runs code analysis.
- *Automated acceptance test stages* assert that the system works at the functional and nonfunctional level, that behaviorally it meets the needs of its users and the specifications of the customer.
- *Manual test stages* assert that the system is usable and fulfills its requirements, detect any defects not caught by automated tests, and verify that it provides value to its users. These stages might typically include exploratory testing environments, integration environments, and UAT (user acceptance testing).
- *Release stage* delivers the system to users, either as packaged software or by deploying it into a production or staging environment (a staging environment is a testing environment identical to the production environment).

- *Deployment pipeline*- It is also sometimes referred to as a continuous integration pipeline, a build pipeline, a deployment production line, or a living build.
- Whatever it is called, this is, fundamentally, an automated software delivery process.
- This is not intended to imply that there is no human interaction with the system through this release process; rather, it ensures that error-prone and complex steps are automated, reliable, and repeatable in execution.

How to implement deployment automation (best practices when you design your automated deployment process)

1. Use the same deployment process for every environment, including production.
2. Allow anyone with the necessary credentials to deploy any version of the artifact to any environment on demand in a fully automated fashion.
3. Use the same packages for every environment.
4. Make it possible to recreate the state of any environment from information stored in version control.

Ansible for configuration management

- Ansible is the simplest solution for configuration management.
- Ansible is a configuration management platform that automates storage, servers, and networking. When you use Ansible to configure these components, difficult manual tasks become repeatable and less vulnerable to error.
- It's designed to be minimal in nature, consistent, secure and highly reliable, with an extremely low learning curve for administrators, developers and IT managers.

- Ansible consolidates resources across multiple systems to manage them from a single platform.
- A configuration management system like Ansible is made up of several components. The systems that are managed can include servers, storage, networking, and software.
- Ansible configurations are simple data descriptions of your infrastructure (both human-readable and machine-parsable) - ensuring everyone on your team will be able to understand the meaning of each configuration task. New team members will be able to quickly dive in and make an impact. Existing team members can get work done faster

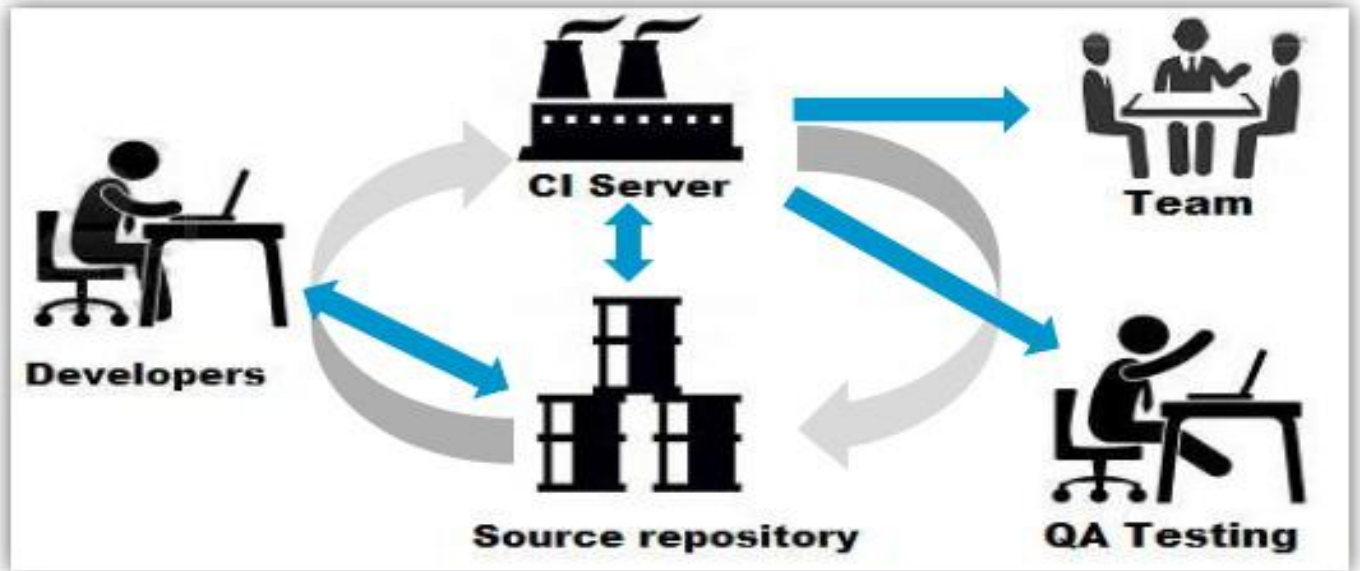
Test automation (as part of CI)

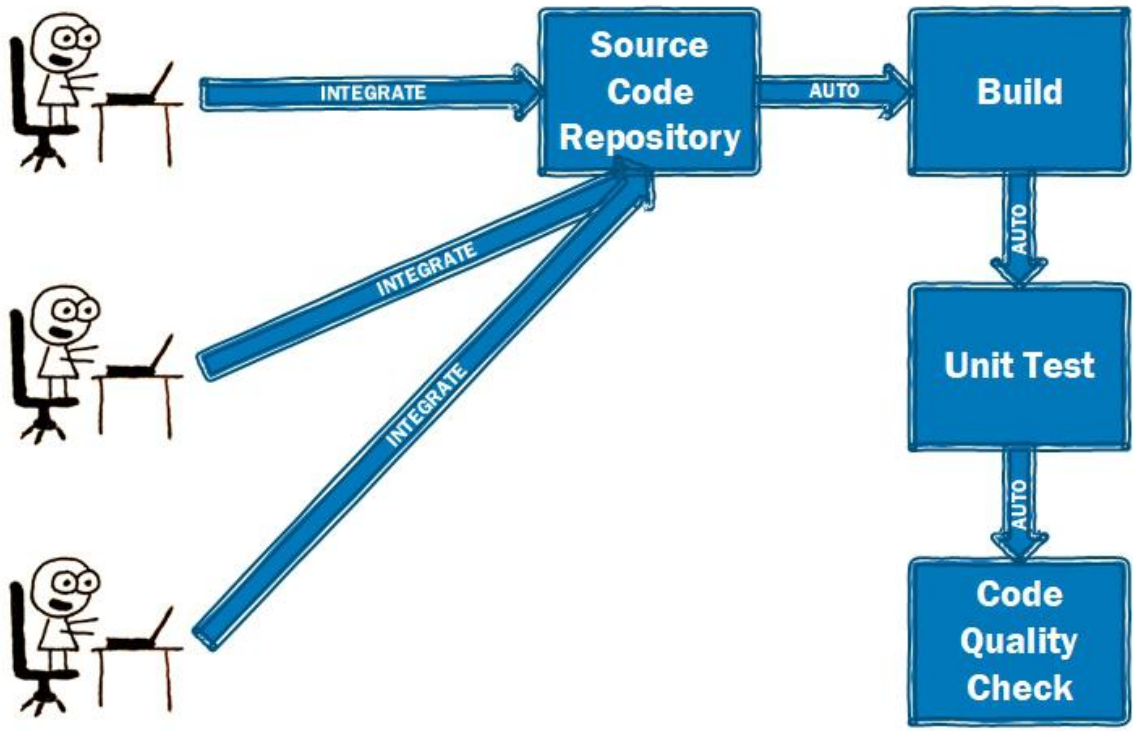
- **Continuous Integration** (**CI**) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests.
-

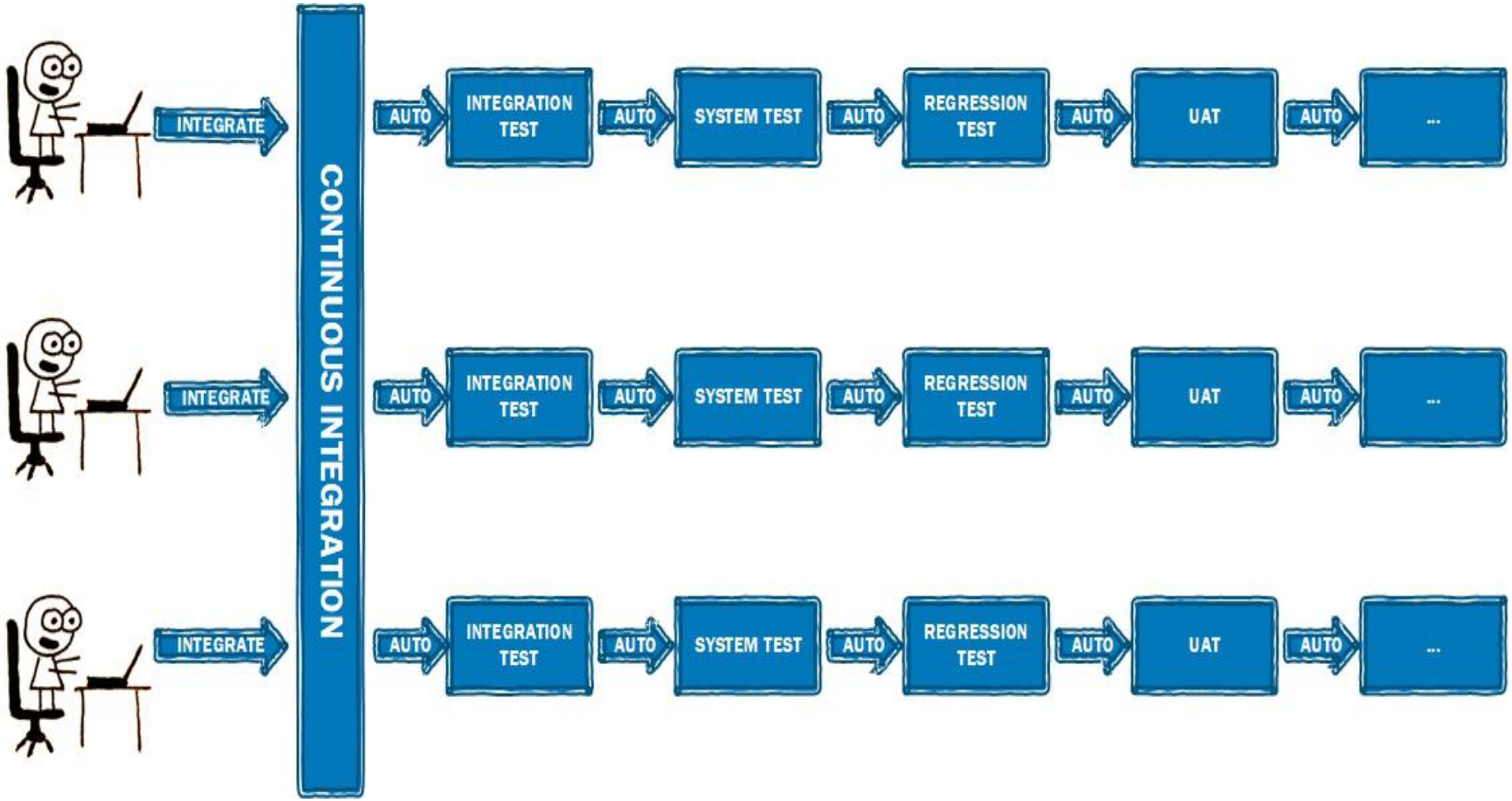
Continuous Integration & Automation Testing

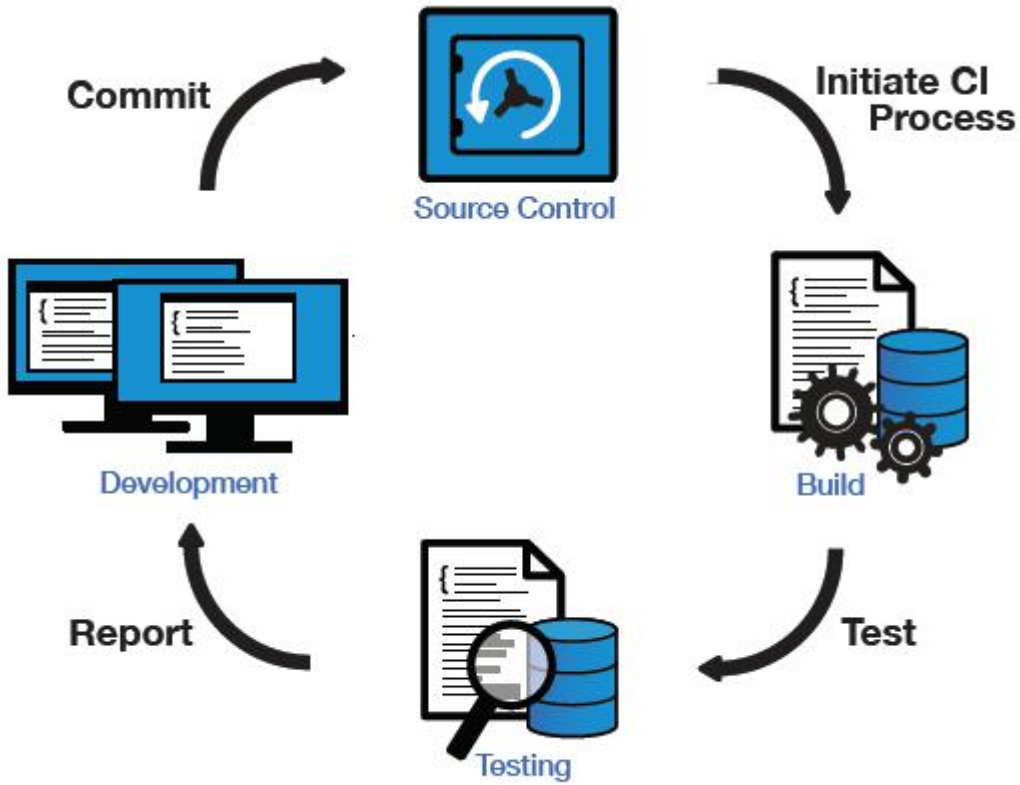
CI Process

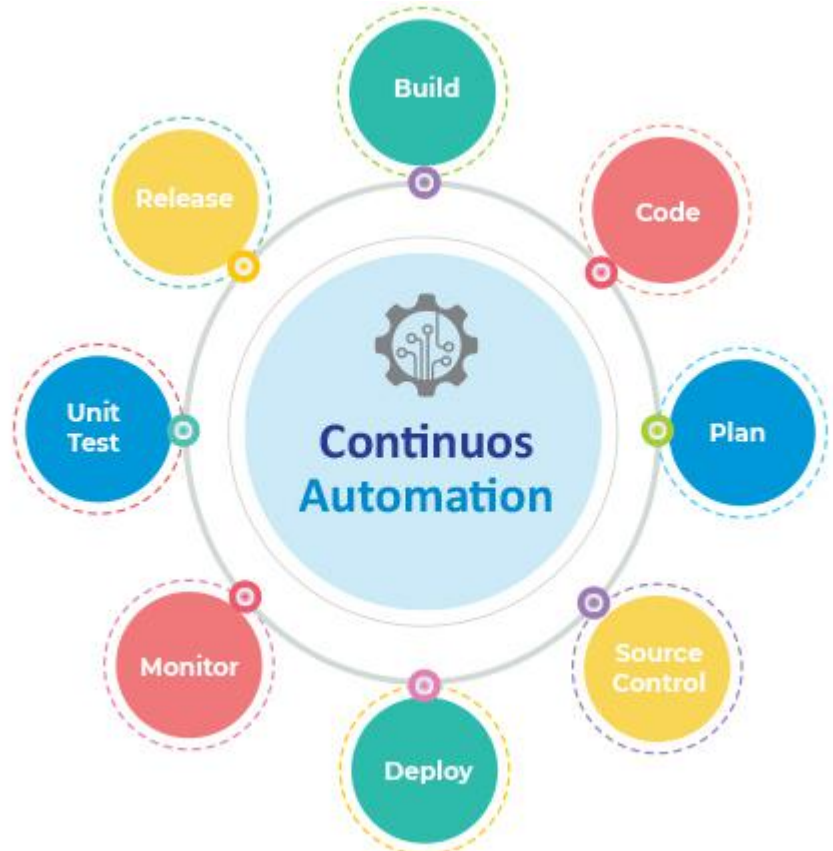
- **Continually integrate and test through the project lifecycle**



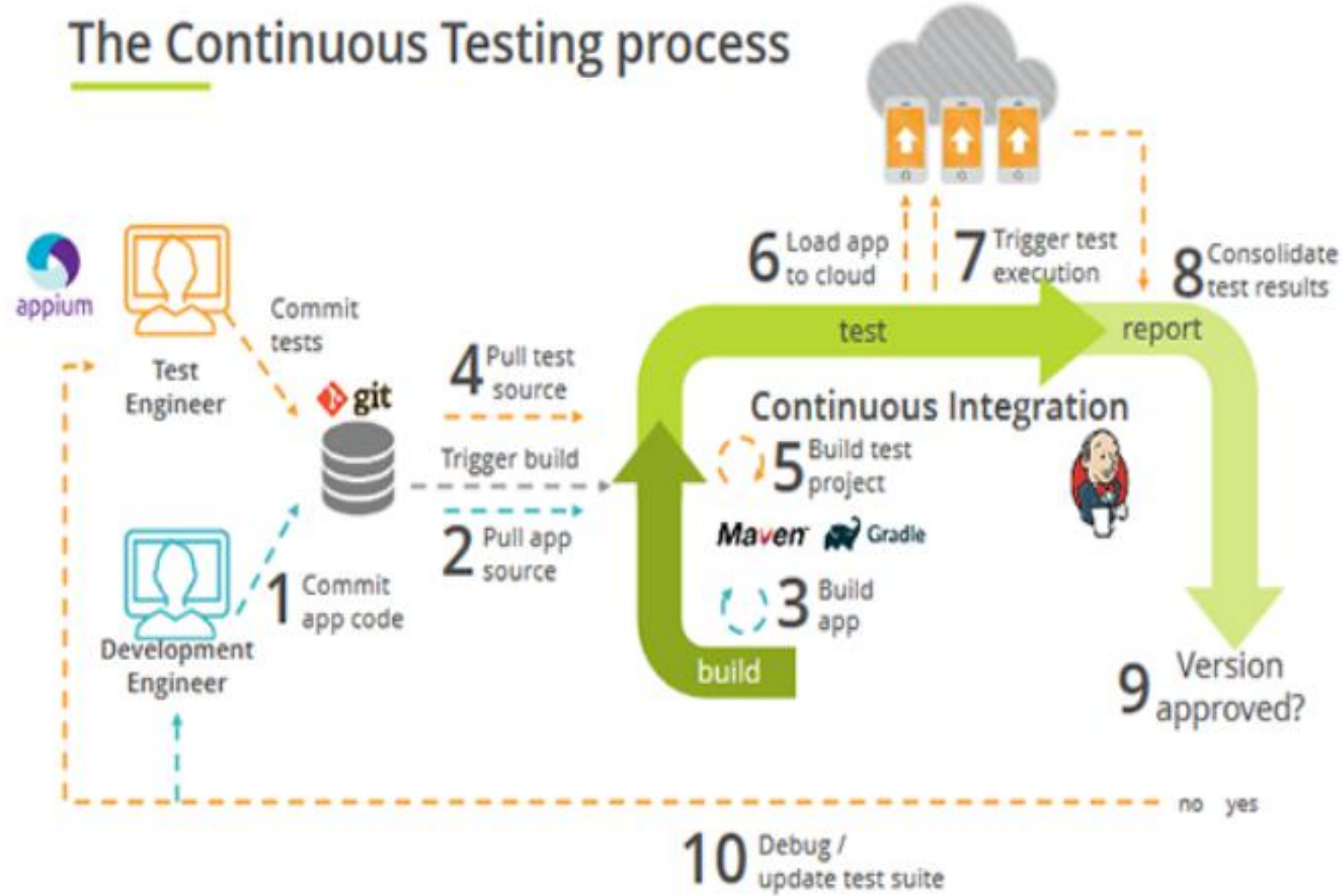

INTEGRATE
Source Code Repository
AUTO
Build
AUTO
Unit Test
AUTO
Code Quality Check
INTEGRATE
INTEGRATE


INTEGRATE
INTEGRATE
INTEGRATE
CONTINUOUS INTEGRATION
AUTO
INTEGRATION TEST
AUTO
SYSTEM TEST
AUTO
REGRESSION TEST
AUTO
UAT
AUTO
...
AUTO
INTEGRATION TEST
AUTO
SYSTEM TEST
AUTO
REGRESSION TEST
AUTO
UAT
AUTO
...
AUTO
INTEGRATION TEST
AUTO
SYSTEM TEST
AUTO
REGRESSION TEST
AUTO
UAT
AUTO
...


Commit
Source Control
Initiate CI Process
Development
Build
Report
Testing
Test


Build
Release
Code
Unit Test
Continuos Automation
Plan
Monitor
Source Control
Deploy

The Continuous Testing process



Robot Framework

- **Robot Framework** is a generic open source automation framework. It can be used for test automation and robotic process automation (RPA).
- Robot Framework is a generic test automation framework for acceptance testing and acceptance test-driven development (ATDD).
- It is a keyword-driven testing framework that uses tabular test data syntax.
- Test cases are written using a keyword-testing methodology written in a tabular format which makes it clear and readable, and conveys the right information about the intention of the test case. For example, to open browser, the keyword used is **“Open Browser”**.

- Robot Framework is open and extensible and can be integrated with virtually any other tool to create powerful and flexible automation solutions.
- Being open source also means that Robot Framework is free to use without licensing costs.
- Robot Framework has easy syntax, utilizing human-readable keywords.
- Its capabilities can be extended by libraries implemented with Python or Java.
- Robot Framework is used very widely around the world in different domains and contexts.

- Robot framework consists of a set of tools, techniques and abstract rules; its job (besides allowing to write automated test cases) is simplifying the test automation process.
- In practice, Robot is a modular test automation framework that has the capability to interact with 3rd party libraries and functions.
- Robot is a test automation framework that uses libraries. With Robot, you can run a variety of automated tests. Selenium is a library (some call it a webdriver).

Architecture:-