**Jenkins Fundamentals**

**Overview**

*Jenkins is an open source automation server that enables developers around the world to reliably build, test, and deliver their software.*

**WHAT IS JENKINS**

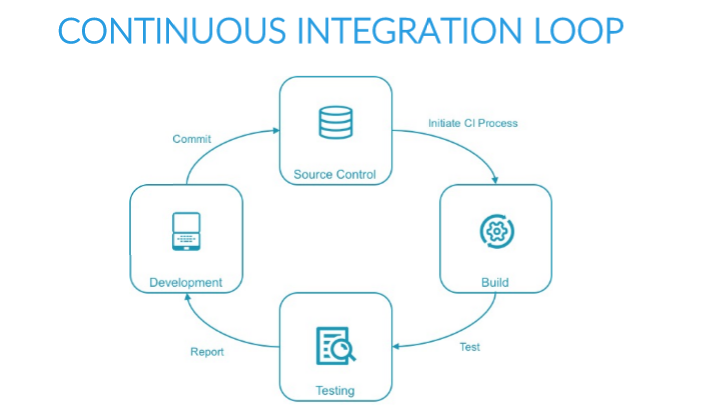
* *#1 Continuous Integration and Delivery server*
* *Created by Kohsuke Kawaguchi in 2006*
* *Original project: "Hudson", renamed "Jenkins" in 2011*

## **IN THE NOT-SO-GOOD OLD DAYS**

* Development, Quality Assurance, and Packaging/Deployment were siloed
* A large group of developers coded for months
* Developers handed the "completed" code to Quality Assurance
* Quality Assurance ran tests and handed bugs back to developers to fix
* Developers fixed the bugs then handed the code back to Quality Assurance,  
  who reran the tests
* Quality Assurance handed the tested code to Packaging/Deployment,
* Packaging/Deployment released the product

## **CURRENT PRACTICES**

* Current practices advocate collaboration between different teams and more flexibility in planning and development and shorter development cycles. Three major philosophies:
* Agile — Emphasizes adaptive planning and evolutionary development. Work is planned and completed in "sprints" (usually 1-2 weeks worth of work), with frequent (usually daily) "scrums" where all team members report progress and plan next steps. See [Agile Manifesto](http://agilemanifesto.org/)
* DevOps — Extends the Agile philosophy into operations and production by advocating for the automation and monitoring of all steps in the development cycle. See [What is Devops?](https://theagileadmin.com/what-is-devops)
* Continuous — Implements Agile and Devops philosophies with tools that thoroughly test each code modification before it is integrated into the official source



## **CONTINUOUS PHILOSOPHY**

* Integrate code often, at least daily, so that integration is a non-event
* Builds are triggered automatically based on **commit** and **merge** actions  
  and the success of upstream builds
* Each integration is verified by an **automated** build (including test)
* Automate the complete build-test-deploy cycle
  + Activities always run in the same order
* Build and test each code modification
  + Find problems sooner, when they are easier to fix
* *Continuous Integration does not get rid of bugs, but it does make them dramatically easier to find and remove.*

## **CONTINUOUS INTEGRATION, DELIVERY, AND DEPLOYMENT**

* **Continuous Integration (CI)** is the frequent, automatic integration of code
* Automatically tests all new and modified code with the master code
* **Continuous Delivery (CD)** is the natural extension of CI
* Ensures that the code is always ready to be deployed
* Manual approval is required to actually deploy the software to production
* **Continuous Deployment** automatically deploys all validated changes to production
* Frequent **Feedback** enables issues to be found and fixed quickly

**Continuous Integration**

* Continuous Integration, in its simplest form, involves a tool that monitors your version control system for any changes. Whenever a change is detected, this tool automatically compiles and tests your application.If something goes wrong, the tool immediately notifies the developers so that they can fix the issue immediately.
* Continuous Integration can also help you to monitor the code quality and code coverage metrics, this encourage developers to take pride in the quality of their code and strive to improve it.
* It has the potential to enable and trigger a series of incremental process improvements, going from a simple scheduled automated build right through to continuous delivery into production.
* A good CI infrastructure can streamline the development
* process right through to deployment, help detect and fix bugs faster, provide a useful project dashboard for both developers and non-developers, and ultimately, help teams deliver more real business value to the end user.
* In essence, Continuous Integration is about reducing risk by providing faster feedback. First and foremost, it is designed to help identify and fix integration and regression issues faster, resulting in smoother, quicker delivery, and fewer bugs.

## **TESTING GOALS**

* Validate that the software meets its goals
* Search for defects that can be fixed to improve software quality
* Facilitate refactoring and upgrades by validating that everything is still working after.

## **TEST ACTIVITIES**

* Good testing involves many different types of tests:
* Unit testing, integration testing, smoke testing
* Functional testing
* Non-regression testing
* Acceptance testing
* Code Quality and Static Analysis
* Performance and Security Testing
* Report test results

## **AUTOMATED TESTING**

* Tests can be run frequently
* Tests should be independent from each other as much as possible
* Tests can be run in parallel
* Tests can be run in the same order
* Consume machine resources but little human time except to review  
  the results of tests that fail
* Running tests frequently means that problems are found early  
  and you usually know what small piece of code caused the problem
* Define different tests to run at different stages of the build chain

## **TYPES OF TESTING — FASTER**

* **Unit tests** — test a small piece of code (a function or method or command)
  + Run blazingly fast, and are often written by the person who wrote the code
* **Integration tests** — validate integration between multiple sub-systems
  + Including external sub-systems like a database
* **Smoke tests** — validates basic functions of the system
  + Also known as **sanity checking**

## **TYPES OF TESTING — SLOWER**

* **Functional tests** — validate the normal software behaviors against  
  the expectations and requirements
* **Non-regression tests** — validates that the system still produces  
  the same result
* **Acceptance tests** — tests the full product from the perspective of  
  the end-user use cases and feelings. Probably includes manual testing

**MANUAL TESTING**

* Is appropriate when test result is subjective, such as user experience testing
* May also be used when the cost of automation is excessive for some reason
* Should be performed rarely, and only on software that has passed all automated tests.