Jenkins

Jenkins : Introduction and some facts

Introduction:

Jenkins is a cross-platform, continuous integration and continuous delivery application that increases your productivity. Use Jenkins to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build.

Facts:

- Written in Java and initially was supposed to be used as a CI tool
- Over 600 plugins to customize Jenkins as per your need
- Over 1000+ public repositories on Github, 500+ contributors, strong commit activity
- Free open source and most widely used tool for maintaining continuous integration cycle. Google trend says it all

Why Jenkins?

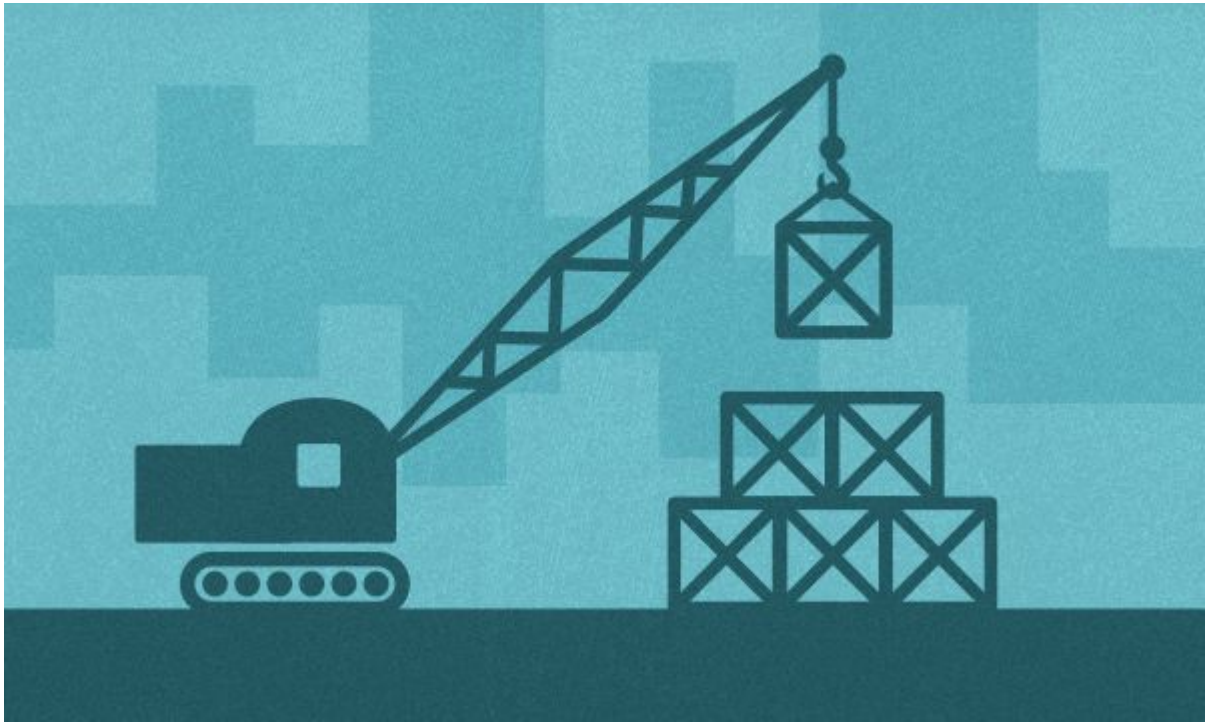
- Easy install, easy upgrade, easy configuration
- GUI to manage Strong community
- Distributed builds – Arguably most powerful feature.
- Monitoring external jobs No limit to the number of jobs, number of slave nodes
- Plugin architecture: Support for various version control systems, authentication methods, notification, workflow building, and many more features can be added.
- Jenkins provides machine-consumable remote access API to its functionalities
- Actually there are lot of useful plugins. The list is too long to mention here. Go on, explore on your own. There’s plugin available for almost everything you would want.

Continuous Integration & Continuous Delivery

- Continuous Integration: It is the practice of merging development work with a Master/Trunk/Mainline branch constantly so that you can test changes, and test that changes work with other changes. The idea here is to test your code as often as possible to catch issues early. Most of the work is done by automated tests, and this technique requires a unit test framework. Typically there is a build server performing these tests, so developers can continue working while tests are being performed.

- Continuous Delivery: It is the continual delivery of code to an environment once the developer feels the code is ready to ship. This could be UAT or Staging or could be Production. But the idea is you are delivering code to a user base, whether it be QA or customers for continual review and inspection.

Automated Build Process



Continuous Integration

- A development methodology
- Of daily developer integrations
- Verified by automated builds
- Every commit triggers a build
    - As soon as you have completed an independent functionality
    - A full build on another, empty machine

Self-testing build

- Directly go from source to running build
  - No manual copying
  - No click on dialog boxes
  - No configuration file editing
- Test with
  - Unit tests
  - Functional tests (web tests)
  - Performance tests
- Responsible persons should be notified when anything fails
- Tests web in more browsers

Continuous Delivery

- Continuous delivery/Continuous deployment
- Continuous, successful methodology to deploying code
- Repeatable and reliable
- Automated the steps of taking checked in code and making it run on production servers, used by customers