[AT Ralph] Continues Integration

Target: Understand Continues Integration related concept and why need it

For Challenge: Setup the Jenkins in your local and run a helloworld job

Raw doc(Chinese version): https://docs.google.com/a/equilar.com/document/d/1djhd6pGO9eMI3ZMk8rAbE16s35mJAqi87siQ0SytFtI/edit?usp=sharing

Part 1: Introduce CI

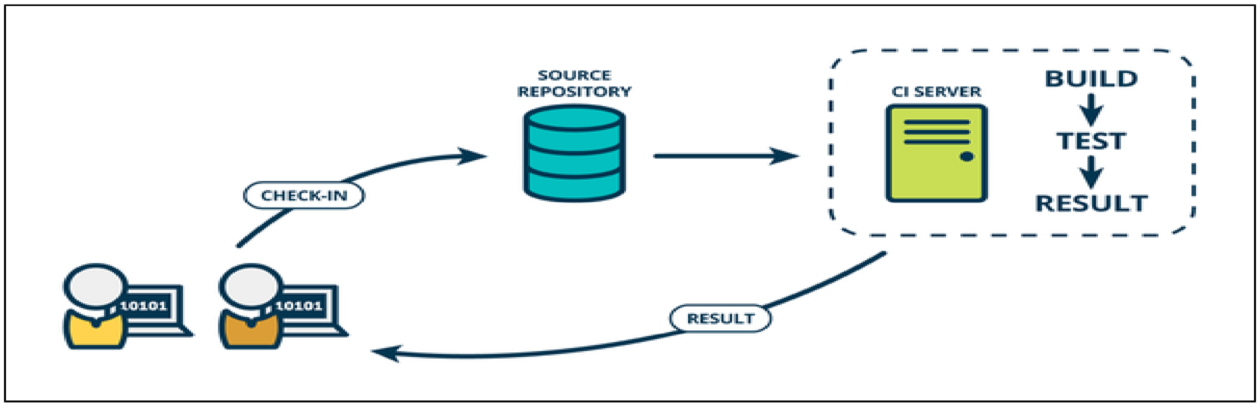
https://www.continuousdelivery.com/

http://www.mindtheproduct.com/2016/02/what-the-hell-are-ci-cd-and-devops-a-cheatsheet-for-the-rest-of-us/

CI (Continues Integration):

CI means that one developer (hi Steve!) who writes code on his laptop at home, and another dev (hey Annie!) who codes on her desktop in the office can both write software for the same product separately, integrate their changes together in a place called the source repository. They can then build the combined software from the bits they each wrote and test that it works the way they expect.

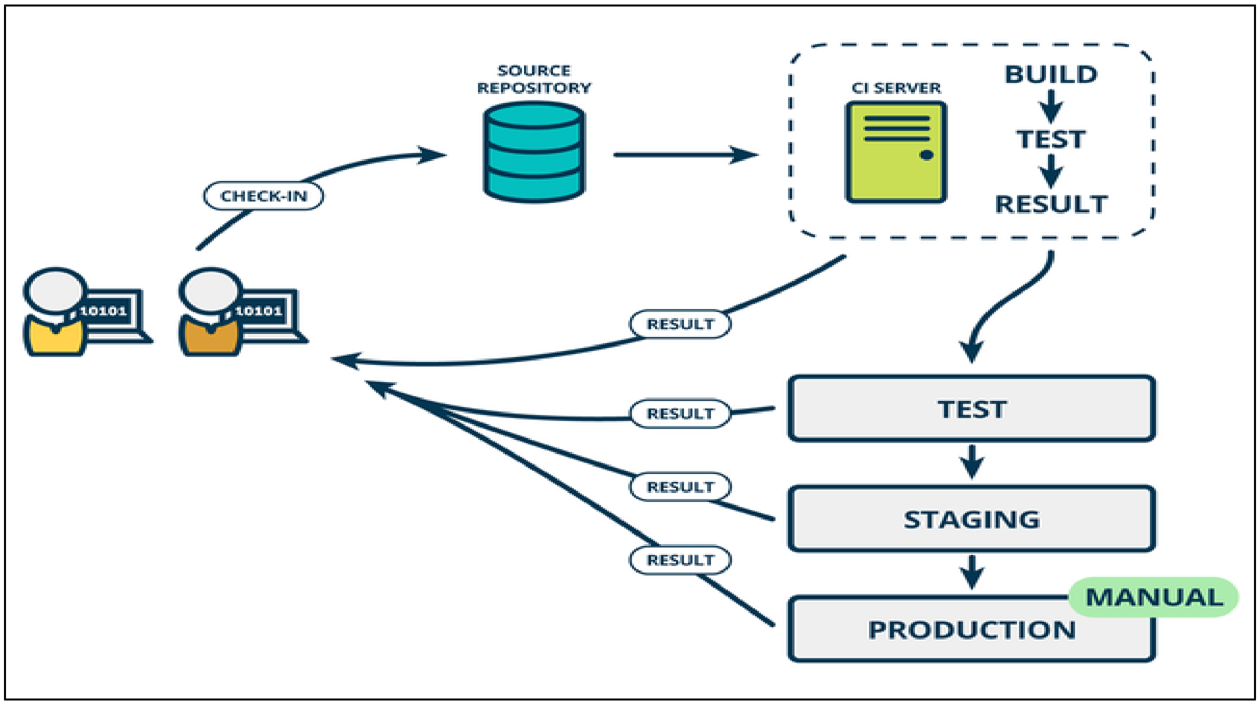
(Develop - Build - Unit test)



CD (Continues Delivery):

Continuous Delivery means that each time Steve or Annie makes changes to the code, integrates and builds the code, that they *also* automatically test this code on environments that are very similar to production. We call this progression of deploying to – and testing on – different environments a *deployment pipeline*.

(Develop - Build - Unit test - Accept test - delivery to QA)

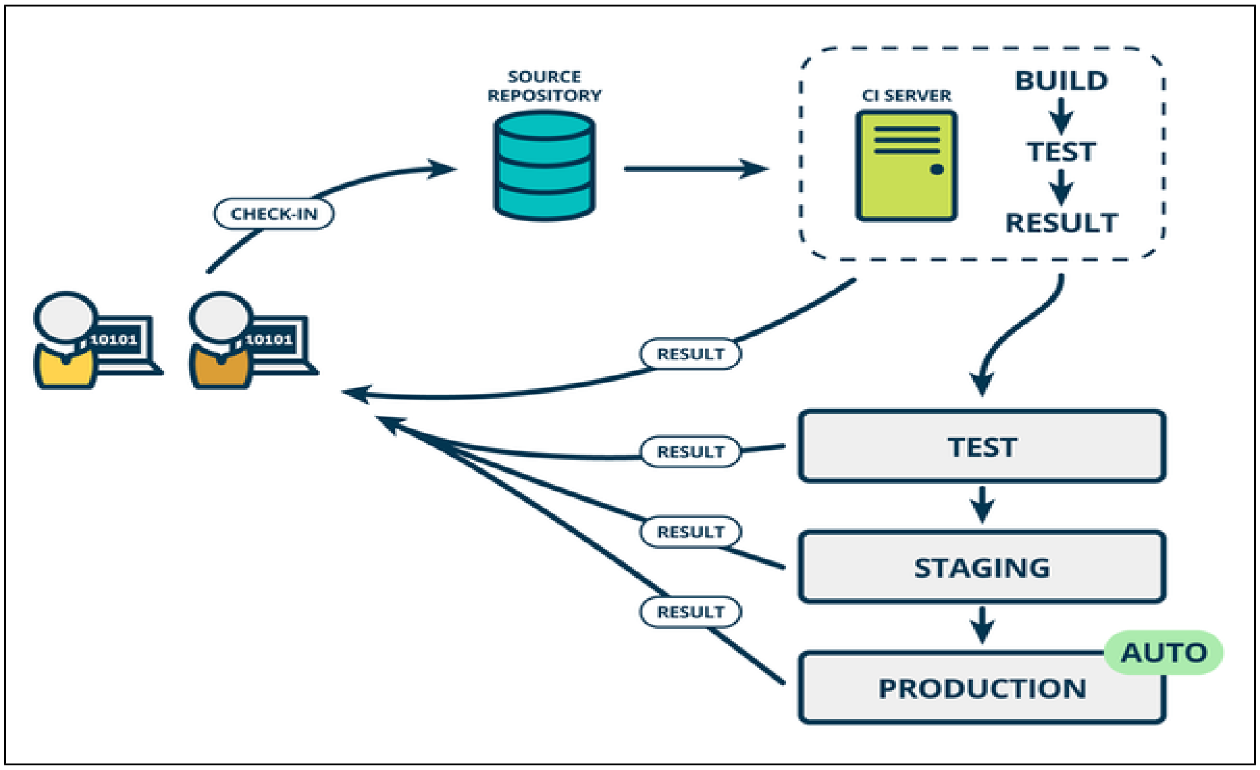


Continues Deployment:

Every change that Steve or Annie makes, and which passes all the test stages, *automatically* goes to production.

(Develop - Build - Unit test - Accept test - delivery to QA - delivery to production)

tar filename.tar * , deployment to production, AnsibleChefPuppet



Part 2: History Of CI

Wiki - https://en.wikipedia.org/wiki/Continuous_integration

Brief History Of Continuous Integration - http://blog.typemock.com/2013/10/a-brief-history-of-continuous-integration.html

In 1994, Grady Booch used the phrase continuous integration in Object-Oriented Analysis and Design with Applications (2nd edition)[5] to explain how, when developing using micro processes, "internal releases represent a sort of continuous integration of the system, and exist to force closure of the micro process." In 1997, Kent Beck and Ron Jeffries invented Extreme Programming (XP) while on the Chrysler Comprehensive Compensation System project, including continuous integration.[6] Beck published about continuous integration in 1998, emphasising the importance of face-to-face communication over technological support.[7] In 1999, Beck elaborated more in his first full book on Extreme Programming.[8] CruiseControl was released in 2001.

From Hudson to Jenkins—A Short History - https://www.safaribooksonline.com/library/view/jenkins-the-definitive/9781449311155/ch01s04.html

Jenkins is the result of one visionary developer, Kohsuke Kawaguchi, who started the project as a hobby project under the name of Hudson in late 2004 whilst working at Sun. In 2009, Oracle purchased Sun. Towards the end of 2010, tensions arose between the Hudson developer community and Oracle, initially triggered by problems with the Java.net infrastructure, and aggravated by issues related to Oracle's claim to the Hudson trademark. In January 2011, the Hudson developer community decisively voted to rename the project to Jenkins. They subsequently migrated the original Hudson code base to a new GitHub project and continued their work there.

Part 3: Why Need CI

Agile development also fast the CI develop, for detail, please check: [AT Ralph] Hello Automation

"Short Cycles that are test-driven and feedback-driven, yielding constant improvement."

Pros:

In <Code Complete> Steve McConnell mentions CI advantage:

1. *easy to locate the error*
2. *early system arhivement*
3. *improve the control to the progress*
4. *improve the customer relation*
5. *fully test each component of system*
6. *build the whole system in shorter period*

can also increase the confident of the team.

Cons:

1. The huge project hard to integrate
2. Human factor
    a. CI keep connection to version control for update
    b. download new project version from version control
    c. automatically compile
    d. automatically test
    e. automatically code analytics
    f. generate run able code for deployment

is it looks beautiful, but in fact need human to communicate
3. The test environments are different from staging and production(docker)
4. More dependency we rely on (amazon s3)
5. Migration costs, no one knows what happened

Part 3: CI tools

CI is not a single tool, they are a bundle of tools

http://www.ruanyifeng.com/blog/2015/09/continuous-integration.html

Popular CI tools:

- Jenkins
- Travis
- Codeship
- Strider
- GitLab CI - GitLab
- TeamCity - JetBrains
- Bamboo - Atlassian

Jenkins and Strider are open source software, Travis and Codeship are free for open source project.

CI functions:

1. support for source code control system, such as Git, Subversion, TFS, able to automatically trigger the following steps after code change. Such as build, test, deploy, etc.
2. support for dependency management tools, such as Java Maven, NodeJS NPM, Ruby Gem, .Net Nuget, etc.
3. support for multiple type of test framework, the early CI only support unit test, or check for single object/component, then added the support for integration test. However, it still not able to verify the system as user expect, so the modern CI tool start to support the functional test, such as UI automation test Selenium, etc.

Other function and background:

The CI run in cloud

The CI run in mobile

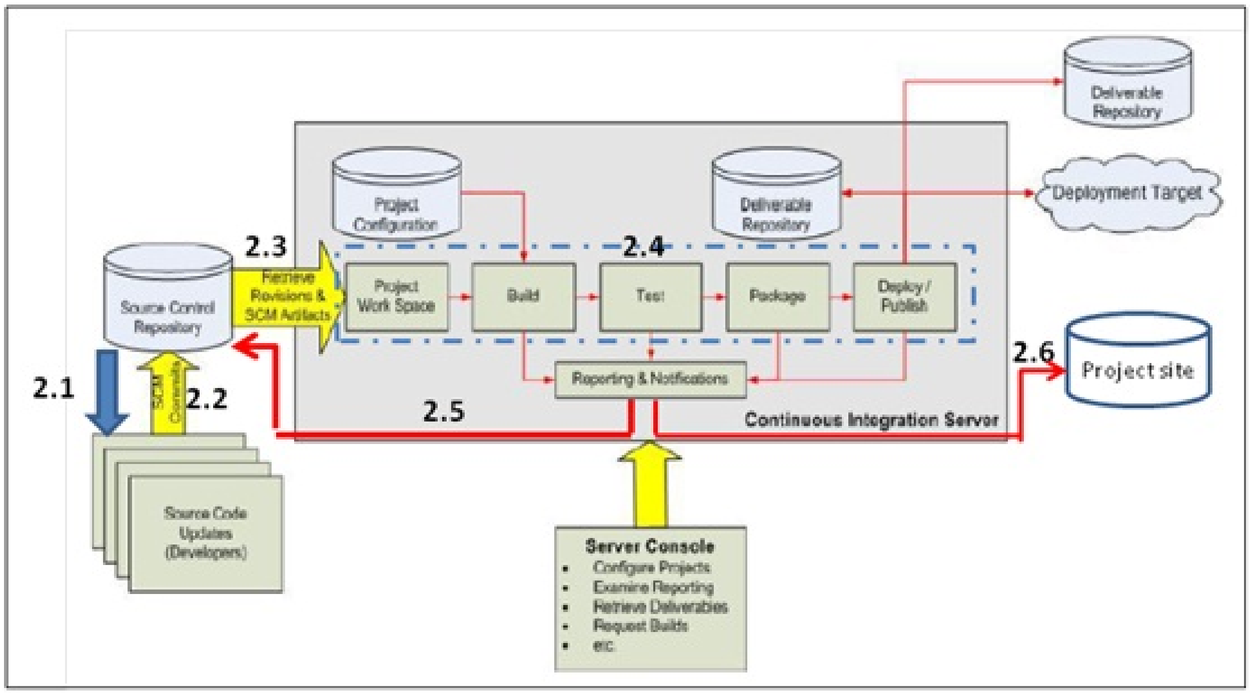
Part 4: Principle Of CI

You can simply image this is windows scheduler or crontab.
Basically it just combine of lots of scripts, if you can run in your local, you also suppose to able to run it in CI. However, it is more functional than windows scheduler and crontab, integrate more plugins, and support permission control and clause nodes. You can add as more node as you want as slaver. Run the job in the time you scheduled or triggered by other program.

Image Jenkins is a super manager, don't need to work himself, but need to coordination each team. The work is done by each team, maven team build, email team send email notification, git team provide source code, etc.

The Key technology for Jenkins: nodes, plugin, credentials control, user management, visualize, simply, robust

How A Continues Integration System Works:



Pre - Condition:

1.1. Create the source code warehouse through version control/config management tools (such as SVN or Git)
1.2. Create related Configuration tools by using building tools (such as Maven pom.xml)
1.3. Create a job in CI server(Jenkins etc.), connect with version control and build tool, config the building trigger

Now is the CI workflow:

2.1. Developer download the lastest source code from source code wearhouse

2.2. Developer write code, test cases, push the commit to version control

2.3. CI server according to the trigger condition, pull the lastest code in version control wearhouse, put in CI build workspace

2.4. Build tool(mavne) start to compile, test, package the code. If neccessary, deploy, release the product

2.5. Through build tool and version control tool, manage the product (send notification or email)

2.6. Establish, manage project develop site automatically

Part 5: DEMO

1. *use insight_perf_benchmark as example, show how a jenkins job is being config*
2. *build a standalone jenkins in your local, run a windows script*

Practice 1:

Download Jenkins from https://jenkins.io/, install in your local, create a helloworld jenkins job.

The job should able to print "Hello world" in console, make sure the job run successfully with blue light.