IJCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Volume-4, Issue-4 E-ISSN: 2347-2693

Continuous Integration and Deployment Modern Technique's

Vivek Verma[1*] and Vinay M[2]

[1,2] *Department of Computer Science, Christ University, Bengaluru India,*

www.ijcseonline.org

Received: Mar/26/2016 Revised: Apr/04/2016 Accepted: Apr/17/2016 Published: Apr/30/ 2016

***Abstract*—** In current world where software companies are moving towards sustainable rapid development and deployment model, it's very important to automate the process of software development, build, testing and deployment to avoid the delay in software release. In software development process many developers are involved during software product development. It is very significant that there should be framework which must notify the compilation and build error at least once is a day, so that reported error can be corrected. In most of the cases developer writes a unit test case to test their own written method. So as soon as new code stored into the shared repository there should be a way that we can perform all the unit test cases execution automatically and publish the result to all the developers. The next part of the problem domain is, how fast we deploy the newly build product version on the test environment and execute the test automation suite and publish the result to the all-stake holder on the new build. The last part of the problem domain is as soon as product got passed from the test environment, it must move to the next (Staging / Production like environment) automatically where again we must perform the basic sanity testing and on successful result framework must deploy the product finally to production environment automatically.

***Keywords*—** Continuous Integration ;Continuous Deployment; Software Development; Regression Testing; Code Coverage, Unit Test; Shared Repository.

I. INTRODUCTION

Continuous integration and deployment both are essential for successful software deliverables. Continuous integration helps us to identify the software code integration issues as early as possible in SDLC to avoid last minute integration conflict [1]. Continuous integration allows the system to build the code at least daily once in order to avoid compilation and other build issue. Continuous deployment help us to deploy the executable file and execute and verify the regression test cases(older functionality) doesn't get affected by the new code and provide the daily execution report with all the stake holder. Its helps us to get confidence on the software release made by the system [6].

A. Continuous Integration

Continuous integration allows the developers to test their code multiple times, at least once in a day, to verify integrity issue. To enhance the result of continuous integration we have to follow the best way of doing continuous iteration [1].

- Every developer has to check in their code every day in source code repository [8].
- Developer has to write unit test cases before cheek in the code to central repository [2].
- Every day at night system has to check out the code from source code repository [6].
- Analysis of the source code to be done using static analysis tool to figure out the Error's in early stage. (For Example Sonar Cube).
- Check the overall code coverage using code coverage tool after unit testing (For Ex Jacaco) [7].
- Use the code coverage tool to figure out how much % of newly added lines are covered using unit testing [7].
- Code coverage helps the tester to figure out the untested part of the code.
- Share the result of static analysis tool and code coverage tool with all stake holders.
- Build the deploy-able file (for Ex jar/war/rpm) from source code.
- Deploy the created deploy-able file into test environment and do the required configuration.
- Perform the Sanity check first before starting Functional test.
- Automatically report the bug in case of Failure [1].
- Automate as many as functional test cases to perform testing automatically [2].

B. Continuous Deployment

In the deployment process, if deployment job is no automated we faced major issue while deploying the software artifact to QA, staging and production environment.

One of the most frequent issue faced is that if software is working in development environment but not working on QA environment , sometimes if it's working on QA environment then it's not working on either staging or

production environment , because some of the steps missed during deployment or because of the deployment read me is not proper [2].

To overcome this issue continuous deployment will help to identify the deployment related issue early either in Development or QA environment, because in continuous deployment process we build, deploy and test the software artifact every day on QA environment to detect and fix the issue as early as possible.

As continuous deployment is an fully automated process there is very rare chance of manual mistake due to which deployment get effected also continuous deployment provide the fastest turnaround time to deploy the artifact to production environment which is the demand of agile methodology development, where every 15 or 30 days software build is out for the use of end customer [1].

II. CONTINOUS INTEGRATIONS AND CODE COVERAGE

Code coverage is one of the ways to check how much testing is performed on the product and how much testing still can perform before releasing the product to production environment.

In code coverage, we generally figure out how many lines of code is covered during unit and functional testing and how much line still not covered, which gives the clear idea to all the stake holder to know which part of the code is required more testing.

There are various open source tools are available for performing the code coverage which can be easily integrated with the continuous integration software Hudson/Jenkins and provide the standard HTML based report for code coverage achieved by the testing.

Project Coverage summary

Name	Classes	Conditionals	Files	Lines	Packages
Cobertura Coverage Report	45% 23/51	74% 469/630	45% 23/51	28% 1450/5222	88% 7/8

Coverage Breakdown by Package

Name	Classes	Conditionals	Files	Lines
Stop-tabac	0% 0/1	N/A	0% 0/1	0% 0/5
Stop-tabac.Classes	100% 1/1	47% 8/17	100% 1/1	27% 58/213
Stop-tabac.Classes.Controller	19% 3/16	73% 22/30	19% 3/16	5% 120/2665
Stop-tabac.Classes.Manager	100% 3/3	63% 20/32	100% 3/3	65% 103/158
Stop-tabac.Classes.Model	75% 6/8	81% 249/306	75% 6/8	77% 669/869
Stop-tabac.Classes.Service	56% 5/9	69% 163/235	56% 5/9	47% 388/830
Stop-tabac.Classes.Utils	60% 3/5	N/A	60% 3/5	59% 74/126
Stop-tabac.Classes.View	25% 2/8	70% 7/10	25% 2/8	11% 38/356

In the above figure we have shown the simple code coverage report where out of 51 classes of code 23 classes got covered using testing; Henceforth the code coverage for classes are 45 %.

III. CONTINOUS INTEGRATION AND PRALLEL TESTING

Continuous Integration helps the tester to test the software in different environment parallel. For Example if tester wants to test a mobile app with different android OS version can be easily done using continuous integration.
Continuous integration tool Hudson supports the pipeline job which can be used to run the different test on the same software build which can be either run sequential or parallel best on the pipe line configuration.

IV. CONTINOUS INTEGRATION AND PRALLEL TESTING

Continuous Integration helps the tester to test the software in different environment parallel. For Example if tester wants to test a mobile app with different android OS version can be easily done using continuous integration.
Continuous integration tool Hudson supports the pipeline job which can be used to run the different test on the same software build which can be either run sequential or parallel best on the pipe line configuration.

V. POPULAR TOOLS

Existing tools which are getting popularity for managing Continuous Integration are:

- Hudson/Jenkins.
- GIT
- Phabricator
- Ant/Maven.
- Jacaco/Cobetura

A. Hudson/Jenkins

Hudson is a continuous integration (CI) tool written in Java which provide the GUI to end user, it can be integrated with source code repository to check out the code from repository, either manual checkout or cron based checkout, also it can be easily integrated with any kind of build script Ant/Maven to build and deploy the software artifact to different environment. It can be easily integrated with any of the scripting language or unit test framework to perform unit and functional testing in automated way. It having their own plug-in to present the HTML based report in to the Hudson fronted.

B. Git

Git is a version control system for software development .It is a distributed revision control system with an emphasis data integrity and support for distributed, non-linear work

flows. In Git we have the main branch where every day continuous integration system checkout the code for build, deployment and testing. Git supports the local branching feature where every developer can create their own branch for development and post development and unit testing he can merge the code with master branch for testing.

C. Phabricator

Phabricator is a suite of web based software development collaboration tools, Phabricator provide the platform to the development team to share the code with his peer members or team lead for review comments before merging the code to master branch, which can be used by the end customer. Also its supports the difference function which can easily figure out the old line and newly added line is the source file for the reviewer.

D. Ant/Maven

Ant and Maven both are developed by Apache and used to build the software artificial for deployment. Both can be easily integrated with Hudson to provide the build and deployment environment.

E. Jacoco/Cobetura

Jacaco and Cobetura both are open source tool for code coverage. They instrument the code during unit and functional testing and show the code coverage achieved from the unit and functional testing.

VI. SUMMARY AND CONCLUSIONS

Our research found that product which is maintained using continuous integration and development:

- Give a clear picture of test coverage with respect to the code coverage.
- Have a past result recorded for comparison.
- Give a clear picture of software quality to all stake holders in terms of graph.
- Reduces the no of production bug.
- Identify the untested part of the code, which need attention.
- Reduces the time to market because most of the old test cases are executed by the system automatically.

Best practices of the Continuous Integration and deployment provide several benefits to the organization, few of the benefits are:

A. Detect Issue early in the SDLC

Most common use of the CI is to identify issues with the software as early as possible. As development team continues to add new features to the code and fix bugs, the code base is constantly in unstable stage. If you are not performing a CI server to continually build your software, Using CI, we build and test the software every day so that we can detect and fix the issue as early as possible.

B. Provide Platform for Continuous Deployment

Continuous deployment is related to Continuous Integration and CI enable the feature to deploy and test the stable build on staging or production environment.

C. Make Software Stable

By continually building the software and (optionally) executing unit and functional tests make the software more stable day by day.

D. Faster Delivery

Continuous integration and deployment both enable the faster delivery of the software artifact to end customer , Also in case of any urgent customer request can be delivered fast using these technology.

REFERENCES

[1] "Testing Extreme Programming", Lisa Crispin and Tip House, 2003, Addison Wesley.
[2] Paul Ammann, Jeff Offutt (2013). Introduction to Software Testing. Cambridge University Press.
[3] Eldh, Sigrid, et al. "Towards a Test Automation Improvement Model (TAIM)." Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on. IEEE, 2014.
[4] Campbell, G., and Patroklos P. Papapetrou. SonarQube in Action. Manning Publications Co., 2014.
[5] Humble, Jez, and David Farley. Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, 2010.
[6] Jenkins https://jenkins.io/doc/, 2015.
[7] Jacoco http://eclemma.org/jacoco/trunk/doc/, 2015.
[8] Git https://git-scm.com/doc , 2015.

AUTHORS PROFILE

Vivek Verma , received the Bachelor in Computer Application degree from Rani Durgavati University, Jabalpur, India and Currently Perusing a Master of Computer Science and application degree from Christ University Bangalore, India. Currently, he is working as a Senior Member of Technical Staff at Oracle India Pvt Ltd

Vinay M, received the Masters of Computer Application Degree from Indira Gandhi open National Open University, India and a Mphil in Computer Science from Madurai Kamaraj University, India respectively. Currently he is an Assistant Professor at Christ University Bangalore, India.