ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

Continuous Integration Tools, Practices and Principles: Review

Sarika Chaudhary

Assistant Professor, Dept. of CSE/IT, Amity School of Engineering & Technology, Amity University, Haryana, India

ABSTRACT: Continuous integration (CI) is basically a practice that can integrate and automate the code several times a day whenever there is any change in code or in the version of software. So on deliver updated version quickly and dependably continuous integration (CI), continuous delivery and deploy is managed. Therefore to implement CI technique it's necessary to possess entire information on numerous tools, challenges and approaches of CI. In this paper various tools and approaches that facilitate CI are reviewed and it is concluded thatcontinuous integration provide a quick feedback in case a defect is introduced into the code and to identify it and resolve it as soon as possible. CI is becoming an important area in the field of software engineering.

KEYWORDS: Continuous integration; delivery; deployment; Software engineering; Tools.

I. Introduction

Continuous integration (CI) is practice in software engineering that merge the working copies of all developers to a common repository many times a day. Grady Booch first proposed CI in 1991 and this concept is adopted by extreme programming. Initially the concept of integrating the working copies several times a day is not taken into account but extreme programming provide the accountability of integration process. Whenever the source code is changed, it is automatically updates [9]. This process of automated update consists of compile, testing, program code inspections, and deploys and is carried out with the continuous integration tools. CI reduces software failure risks, repetitive processes, produces high quality product that provide better scalability, flexibility, increased project visibility, and finally strengthened the confidence in the operational team and development team [2].

According to Muhonen [6], to take a fast feedback on SDLC methods CI are often accustomed mechanically integrate useful testing. CI advocates desegregation work-in-progress multiple times per day, CDE and CD area unit concerning ability to quickly and dependably unharness values to customers by conveyance automation support the maximum amount as attainable [2], [6].

As growing range of commercial cases indicate that the continual practices area unit creating inroad in package development industrial practices across numerous domains and sizes of organizations. At identical time, adopting continuous practices isn't a trivial task since structure processes, practices, and power might not be able to support the extremely advanced and difficult nature of those practices[3][7].

According to Martin Fowler [8], CI may be a apply of software system development wherever the team members desegregation their work as usually as potential, typically the team members desegregation their works, daily. Every of integration are verified by automatic build that contains testing that facilitate distinctive the failures of integration in early. Continuous integration apply is taken into account by several groups that it will scale back the matter of software system integration considerably [2][4][7]. Integration may be one in all the last innovate software system development comes wherever all of the various unit are merge along and integration take a look at for verificatory the unit whether or not they interacted one another as planned [6].

Continuous Delivery (CDE) is aimed toward making certain an application is usually at production-ready state when success-fully passing automatic tests and quality checks. CDE employs a collection of practices e.g., CI, and deploying automation to deliver software package mechanically to a production-like setting [10]. According to [6]

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

and [3], this practice offers several benefits such as reduced deployment risk, lower costs and getting user feedback faster.

Continuous Deployment (CD) practice goes a step more and mechanically and ceaselessly deploys the appliance to production or client environments. There's strong dialogue in educational and industrial circles concerning denying and identifying between continuous deploying and continuous delivery [4], [5], [9]. What differentiate continuous deploying from continuous delivery could be production surroundings i.e., actual customers. It's necessary to notice that CD apply implies CDE apply however the converse isn't true [1]. Whilst the final deployment in CDE is a manual step, there should be no manual steps in CD, in which as soon as developers commit a change, the change is deployed to production through a deployment pipeline. CDE practice is a pull-based approach for which a business decides what and when to deploy; CD practice is a push-based approach [5]. In different words, the scope of CDE doesn't embody frequent and automatic unharness, and CD is consequently a continuation of CDE. While CDE apply is applied for all kinds of systems and organizations, CD apply could solely be appropriate for sure kinds of organizations or systems [2], [7], [8].

II. CI TOOL OVERVIEW

A. *Jenkins*

Jenkins is an open source CI tool and is developed in Java. It can build and test software projects continuously and monitor external environment. Jenkins supports an array of SCM tools—Git, Mercurial, Subversion, Clear case, and many more. We can build Apache Ant and Apache Maven based projects and other shell scripts or Windows batch files for pre- and post-build activities. Note that almost all the configurations can be done via the web-based GUI[11][12].

B.*Buildbot*

Developed in Python, Buildbot is based on the twisted framework. It started as an alternative to the Tinderbox project and is now used in Mozilla, Webkit, Chromium, and others. Buildbot installation has one or more masters and a collection of slaves. The masters monitor source code repositories for changes, coordinate the activities of the slaves, and report the results to users and developers. Slaves run on a variety of operating systems. We need to provide a Python configuration script to the master for the Buildbot configuration. This may be a little difficult for non-programmers to manage, but such scripts give Buildbot much-required flexibility. Buildbot's design allows installation to grow with requirements, beginning with simple processes and growing to meet unique needs[4].It can also be used in handling cloud computing issues effectively[13].

C. *Travis CI*

Travis CI is probably one of the easiest CI servers to get started with. Travis CI is open source and obviously free to host on your own server, but it also offers a SaaS version that allows free testing for open source projects[3][4]. Setup is as easy as linking your GitHub account, giving the relevant permissions, and updating the travis.yaml file with your project specific requirements. A new Travis CI build is triggered after a file is committed to GitHub.

E.*Strider*

Strider is written in Node.JS and JavaScript, and uses MongoDB as a backing store. MongoDB and Node.js are prerequisites for installing Strider. However, Strider is extremely customizable through plugins and may require you to put your hands in code not a bad thing to do, but if you would like a quicker setup without much programming effort, you should probably look at other options[6].

F. *Go*

With Go, regularly performed tasks can be added as pipelines. The instances of these activities are called jobs. Another interesting addition is the ability to visualize the entire continuous delivery workflow with the value stream map. The map helps you track the entire change from commit to deployment. It was created and then open sourced by Thought Works. As with other advanced CI servers, Go lets you distribute your builds across different systems and monitor them all in one place. To use this, you need to install it on your server there is no SaaS available[5].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 4, April 2017

G.*Integrity*

Built on Ruby, Integrity needs Ruby 1.8.7 or newer, Ruby Gems 1.3.5 or newer, and Git 1.6 or newer. There's no SaaS available and need to install it locally before using it. The configuration is done using the "init.rb" file. A sample of this file is available on the project page. Another important thing to note is that Integrity currently works with Git only, so if you use another SCM tool this may not be right for you[3].

H. *Bamboo*

Bamboo is one of Atlassian products that support CI. It can be installed on a local machine or over the Internet. These tools are semi-paid, because it provides a trial in a few days. For GUI testing, this tool does not provide a plugin for using tools of SikuliX and JAutomate. But, there are several guides on the internet which makes Bamboo possible to perform GUI testing. It needs a deep investigation for implementing GUI testing in Bamboo with windows environment[10].

III. CI PRACTICES AND PRINCIPLES

Continuous integration, the apply of oftentimes integration one's new or modified code with the present code repository ought to occur oftentimes enough that no intervening window remains between commit and build, and such no errors will arise while not developers noticing them and correcting them instantly [10] Normal practice is to trigger these builds by every commit to a repository, rather than a periodically scheduled build. The practicalities of doing this in an exceedingly multi-developer atmosphere of fast commits area unit specified it's usual to trigger a brief time once every commit, then to start out a build once either this timer expires, or once a rather longer interval since the last build. Many automated tools offer this scheduling automatically.

Another issue is that the want for a version system that supports atomic commits, i.e. all of a developer's changes could also be seen as one commit operation. There's no purpose in making an attempt to create from solely 1/2 the modified files.To achieve these objectives, continuous integration relies on the following principles [7][9].

Build a code repository: This practice advocates the use of a revision control system for the project's source code. All artefacts needed to make the project ought to be placed within the repository. Program code repository are maintained by the version control system tools that may facilitate the developers to commit the modification of program code, revert to the previous version of program code, and merging the road of codes that is conflict.

A.*Automated build:* All of the build method ought to be worn out automatic manner, with a straightforward method, that isn't needed, the user interaction[10].

B.*Automated testing:* The build process should consist of testing that utilizes a unit testing set.

C.*Code commit:* The program code should be committed by each of developer to the repository of main every day after they made change[10].

D. *The time of build:* For the effectiveness of the build process, it should all done automatically and does not require a long time. CI build should not be longer than 10 minutes [10].

E.*Feedback:* In the end, it is important for all team members for getting feedback from integration process.

IV. CONCLUSION AND FUTURE WORK

Continuous integration (CI) could be a follow which will modify the build once the source code file is modified. Every build consists of compile, testing, program code inspections, and deploy. Machine-driven build follow is surpass victimization the continual integration tools. The good thing about continuous integration practices i.e. reducing the probabilities of computer code failure risks, reducing repetitive processes, produces computer code merchandise that able to deploy, offer higher project visibility, and increase confidence within the team.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering
(An ISO 3297: 2007 Certified Organization)
Website: www.ijircce.com
Vol. 5, Issue 4, April 2017

REFERENCES

1. Seppala, A., “Improving software quality with continuous integration,”M.S.thesis,deptComp.Sci.Eng,Aalto Univ,Espoo,Finland,2010.
2. Jovanovic, “Software Testing methods and Techniques,” IPSI BgD Trans,InternetRes, vol.5,no.1,pp30-41,2009.
3. Fewster, G.D., “Software Test automation:effective use of test execution tools,” Addison-Wesley Publishing,1999.
4. Ahmad, S.F., “Test driven Development with Continuous integration: A Int driven Dev. With Cotin,Integer,” a Lit.Rev. vol.2, no.3, pp 281-285,2013.
5. Duvall, G.A., Paul, M. and Steve, M., “Continuous integration: Improving software quality and reducing risk,” Pearson Education,2007.
6. Muhonen, M.,“Software Process Improvement: Continuous Integration and Testing for Web Application Development,” M.S. thesis, Dept.Comp. Sci., Tampere Univ, Tampere, Finland, 2009.
7. Geiger, C., Przytarski, D., and Thullner, S., “Performance Testing in Continuous Integration Environments,” M.S. thesis, Dept. Soft. Tech.,Stuttgart Univ, Stuttgart, Germany, 2014.
8. Fowler, M., Foemmel, M., “Continuous integration,” (Thought-Works) http://www. Thought works. com/Continuous Integration. pdf,2006. [Online].
9. Beck, K., “Extreme Programming Explained,” Second Edition, Addison-Wesley Professional, 2004.
10. Pralienka, F., Muhamad, B.and Sarno,R.,“Visual GUI Testing in Continuous Integration Environment”, International Conference on Information, Communication Technology and System (ICTS),2016.
11. Komal, “Comparative Assessment of Human Intelligence and Artificial Intelligence,” International Journal of Computer Science and Mobile Computing, vol.5,page no. 427-432,2016.
12. Komal, “Cognitive science: bridging the gap between machine and human intelligence,” International Journal of Computer Applications,vol.114,page no 16-19,2015.
13. Narang, A., “A review-Cloud and cloud security,” International journal of Computer Science and mobile Computing, vol.6,page 178-181,2017.

BIOGRAPHY

Sarika Chaudhary is an Assistant Professor in the Computer Science Department, Amity School of Engineering and technology, Amity University Haryana, India. She is currently pursuing Ph.D incomputer science and received Master of Technology(M.Tech CS) degree in 2007 from BANASTHALI VIDAPITH, Rajasthan, India. Her research interests are Software Engineering, Datamining.