IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Continuous Integration and automation

Sikender Mohsienuddin Mohammad

Sr. Associate, KPMG LLP, Department of Information Technology, Wilmington University
419 V street, Apt D, Sacramento, CA USA

Abstract

Current IT strategies depend largely on the ability of companies to push changes/features/fixes in an agile and error-free manner. The need for automation increases dramatically as technology advances. Waterfall software development techniques have been replaced by agile software development approaches in the last decades. In this shift, companies, operations, and software developers are unquestionably relatively close to the shared goals. To promote this merger, the industry offers a large range of open source and proprietary solutions. Technologies have increased themselves and thus new techniques have been developed, namely continuous integration (CI), continuous delivery (CDE) and permanent deployment (CD), assisted by all of the above-mentioned open source and proprietary solution. Automation is the primary requirement for DevOps, and DevOps's main concept is "Automate everything." DevOps automation commences from the coding on the developer machine until the code is run and then the application and system are controlled in production. The whole DevOps pipeline includes continuous integration, ongoing testing, and continuous deployment, including live application monitoring. The main focus of DevOps practice is the design and configuration of automated infrastructure and software delivery. DevOps practice depends heavily on automation, to implement for a few hours, and to execute frequently across different platforms. Automation in DevOps, therefore, promotes speed, greater precision, consistent, and reliable delivery rates. In the end, DevOps automation captures everything from design, delivery, and supervision.

Keywords: Continuous Integration, automation, DevOps, CI pipeline

I. INTRODUCTION

Continuous integration is a concept of coding and practices which leads software developers to introduce minor changes and constantly review code in version control repositories. Since most modern systems involve the development of code on different platforms and technologies, a process is needed to incorporate and verify their changes [1]. Continuous integration is a development method where developers often incorporate code into a fully integrated system. Automated construction and automated tests can then check any integration. Automated testing is usually not a strict part of CI. One of the main advantages of daily integration is that you can easily spot and identify mistakes quickly. Since each introduced change is usually small, you can quickly identify the particular change that has introduced a defect. In recent times, CI has become such a standard protocol and a set of basic elements for software development [1]. Reassessment control, construction automation, and automatic testing are some of the examples. Besides, constant delivery and integration have

been developed to provide best practices to ensure your app is deployable any time or even to automatically put the core codebase into development when new amendments are introduced. This enables the team to move quickly while maintaining high-quality standards that can be automatically monitored. Continuous integration won't get rid of bugs but makes finding and deleting them substantially easier. This essay explores the concept of Continuous Integration and automation and how it makes changes to the different platforms and technologies [2].

II. LITERATURE REVIEW

According to Childs et al., constant integration requires constant testing, as the goal is to provide users with reliable software and code. The automatic regression, output, and other tests performed in the CI pipeline are typically implemented as continuous testing [2]. In the first step of creativity of Agile software development models, a major idea is to iterate configuration changes more rapidly and decide the right direction through experimentation – in theory, to "fail quickly" and to optimize consistency as a basic project goal. The lack of comprehension and failure to predict a client's changing needs contributes to a developer's inadequate knowledge to identify long-term program needs to adequately define long term requirements of a project at the beginning [3].

Agile methodologies also facilitated full-time collaboration with the production team of customer partners, offering in-house, real-time insight on consumer preferences and needs, to create an iterative, fail-fast workflow [5]. Agile methodologies have generated a constant continuous cycle between customer subject and software development teams in real-time. Following this idea, DevOps builds on the real-time feedback loop principle and expands it to other points within the SDLC development process, reducing the risks of linking developers, quality assurance (QA), operators, and developer-to-software disconnects.

III. CONTENT

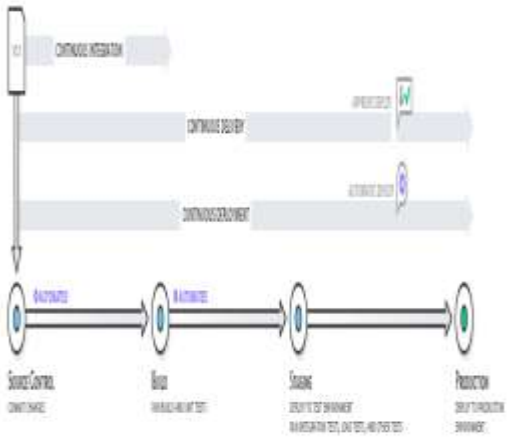

Fig i: The process of continuous integration [6].

A. The need for continuous integration

In the past, a team of developers could work alone for a lengthy time and combine their improvements only after the master branch has been completed. It makes it complicated and difficult to merge code, and it also allowed bugs to build up without correction for a long time [6]. Such factors made it more difficult to quickly provide customers with updates.

B. How continuous integration functions

Application developers devote to a shared repository using a release control system like Git by continuous integration. Before any interaction, developers may agree to run local test scripts on their code as an additional testing layer before integration [7]. A configuration management service automatically constructs and performs unit tests on new code modifications to immediately retrieve errors. The changes made are built, tested, and equipped automatically for retrieval with continued delivery. Continuous deployment broadens the integration process through the installation of all changes made to a test environment or a production environment after construction.

C. How does the CI require?

Many CI vendors say that all their business needs are software and a little set-up to take these development practices. I've got a bridge to give you if you think so. Most companies need the full change in software development culture, adapting their organization, changing workflows, automating the bulk of their tests, and installing significant infrastructure, or learning how to love the cloud [8]. That's not difficult, but it's not easy.

The technological aspects of this change are simpler than the organizational and cultural aspects. For example, it can involve a painful conversion process to put your repositories on Git. However, it is hard to train your developers not only to monitor their work several times a day but also, before the first coding session, to remove any conflicts immediately (when their responsibilities lie) and to write good tests that may be integrated into the automated handling suite [9]. For example, it is simple to use Jenkins to initiate check-in and to report information to developers; it is more difficult for developers to listen to reports promptly and halt new ones to correct old mistakes.

Fig ii: CI model [10].

D. Continuous Integration Tools

1. Jenkins Jenkins

Jenkins is one of the most common, software-based free open source CI solutions. It is a cloud-based, Java-written CI program that includes and running a web server [10]. Thousands of users worldwide like to work with Jenkins because it enables quick building and testing of automated products.

Highlights:

a) Software locally
b) Free of charge
c) Customization of deep workflow
d) Functional and plugin-rich
e) Using OS X, Unix, and Windows packages, the installation is fast [11].
f) Manufacturer for developers
g) A well-established and reputable company

2. TeamCity

TeamCity is a versatile business CI solution for the first hundred configurations that can be used free of charge. One can run parallel constructions with TeamCity simultaneously, mark the constructions and label those hanging. TeamCity is easy to install with a user-friendly interface [12]. It will also support community and professional careers.

Highlights:

a) Free up to 100 setup settings
b) Three runs with three building agents at the same time
c) Can import source code from two different VCS in one compilation
d) Ability to substitute testers with agents
e) Allows improvements to check without VCS commit.

3. Bamboo

Bamboo has a quick and efficient drag and drop user interface, dependent on a server and deploy tool from Atlassian. This tool is popular with developers using other tools from Atlass [13]. Bamboo enables the creation and fusion of new branches after testing automatically. Continuous use and distribution are simple to achieve with this resource.

Highlights:

1) Efficient integration with other Atlassian instruments
2) Free-to-use user interface with drag & drop feature
3) A good method of notification
4) Easy management for company CI scaling.
5) Automation of the test utilizing elastic substances.
6) Automatically recognizes building artifacts by running every pipeline

4. Buddy

Buddy is a DevOps automation tool for ongoing integration, on-going deployment, and reviews. This tool has been developed to operate with Bitbucket and GitHub repository code-based projects [14]. Buddy is a business tool that has a simple, easy-to-use interface and streamlined material design. Customer-orientated service is supported 24/7 by live staff and can be installed on a device in its client version.

Highlights:

a) Instinctive user interface
b) The intuitive design of the deployment flow
c) Aid for docker
d) Presets and tips available
e) Provides sophisticated automation and needs fundamental knowledge
f) Ability to modify the code developed
g) Cloning, variable, relation and note versatile automation [15].

5. GitLab CI

GitLab CI is open-source code and a free continuous integration tool. GitLab API is a highly scalable device and easy to set up and configure for GitLab-hosted projects. Besides testing and creating projects, GitLab CI can implement constructs. This tool identifies areas where the development process needs to be improved. The GitLab developers select individual GitLab CI without giving it a second thought as seamless project integration is automatically achieved [16].

Points to note:

a) Support Docker
b) A quick build server configuration
c) Runs on several machines simultaneously
d) Strong product integration is possible with APIs ready for a range of features
e) A choice to protect confidential project data

6. CircleCI Circle

CircleCI is an integration that is continuous and acts like a delivery network. This can be built locally and used in the cloud, this supports many code languages. This tool facilitates automated testing, construction, and deployment [17]. The basic user interface has many configuration options. Developers can reduce the number of bugs with CircleCI and quickly improve app quality. CircleCI provides a free plan for open-source projects, although this is a commercial tool.

Most important:

a) Aid for Docker
b) Profound adaptation and fast scaling
c) Rich options of integration
d) Sophisticated interface for management
e) A reliable method of building automation
f) Allows complex workflows to be created
g) Enables many builds to run at the same time

7. TravisCI: TravisCI

TravisCI is a demonstrated CI solution that is suitable for open-source projects. This seamless integration platform provides a range of CI automation choices. Since its service is hosting a cloud service, no server is needed [18]. TravisCI is also available on-site in a business-based version. One of the greatest things with this tool is that every time one runs a new one, it supports the latest build.

Most important:

a) Help for many languages and channels
b) Maintenance and statistics of automated deployment
c) Access management for enterprise-grade
d) Optimization Effortless GitHub
e) Study parallel
f) Scaling of demand capacity
g) Support of flow and pull requests for branches

D. CI implementation

When a company practices CI, all of its work is regularly integrated into the main code model (recognized as the trunk, master, or mainline). Research from DevOps Research and Evaluation (DORA) has shown that companies qualify better at least every day when developers fuse their work with the trunk. A series of automated checks are conducted before the actual merger to verify the regression bugs do not occur [20]. If these software products fail, the team usually stops to correct the errors.

All downstream processes should use the packages created by the CI build. These constructions should be numerical and reproducible. At least once a day, you should successfully run your construction process. Automatic testing suite. Start by writing many tests covering the high-quality functionality of your system, if you don't have one. Ensure the check is trustworthy. Users know, that's how when they crash, and when they pass, one must be sure that there are no significant device problems. After this, they make sure all new features are tested. These tests are to be conducted fast to feedback from developers as fast as possible [21]. At least once a day, tests should be successful. In the end, developers will get input from them every day if they have success and acceptance checks. A CI system that performs building an automated check-in test. The system must also visualize the status of the team. Users can enjoy it — and can use horns or stop signs to suggest when the building is broken, for example. Use no email notices; many people ignore email notifications or create a filter hiding notice. Chat system notifications are a better and more popular way to achieve this. Continuous integration, as described by the Kent Beck group and the extreme programming group, often includes two additional activities that also predict the higher performance of software:

Style of construction focused on the trunk in which developers build small lots on trunk/master combine their tasks into a specific trunk/master at least every day, not on long-lasting branches of apps. An understanding that it will postpone all other jobs when construction ends. Automated unit testing is required for CI. These tests should be sufficiently comprehensive to ensure that the software functions properly [22]. The tests also have to take a couple of minutes or less. If automated unit testing lasts longer, developers do not want to run it often. If tests are running uncommonly, the results of many different changes might be a test

failure, which makes debugging difficult. Tests that are rarely done are difficult to maintain.

It is intricate to create maintainable unit test suites. A good solution to this issue is to practice test-driven development (TDD) in which developers are initially failing automatic tests before implementing the test code. TDD provides many advantages: One is that it means that developers write flexible, test-friendly code that reduces the maintenance costs of automated test suites as a result [23]. Many companies do not have maintainable unit test suites and still do not practice TDD, despite this.

E. Objections to CI

CI is often considered a contentious activity, as previously mentioned. CI demands that the developers break up large functions and other modifications into smaller, often integrated incremental changes. This is an improvement for developers who don't have this way of working [23]. Furthermore, it can take longer for teams to transition to small process to complete bigger features. Even so, the frequency over which developers can define a large feature to be completed on a branch is not optimized. One would want to be able to evaluate, incorporate, check, and execute improvements as soon as possible. The effect of this phase is the quicker and more reliable software creation and distribution when changes are small and self-contained. Working in small groups often means that software staff receives daily input from other developers, reviewers, consumers, and automated performance and safety checks on the effect of their work in the system as a whole. In turn, it makes detecting, sorting, and resolving problems easier and faster [24]. Notwithstanding these objections, it should be the number one priority for any organization wishing to begin the journey to continuous integration to assist software development teams.

F. Common flaws

The following are some common limitations preventing broad adoption of CI:

One should not put everything in the repository code. All the applications and the system that are required to build and customize should be in the repository. This may seem beyond the scope of CI, but it is an ideal foundation. The building method is not automated. Manual steps build error opportunities and undocumented steps. Not timely checking for all modifications. Full end-to-end testing is necessary; however, fast tests are also essential (usually unit tests) for quick feedback [25]. Not immediately fix broken builds. One key aim of CI is that everyone can build a stable building. If the construction cannot be fixed in a few minutes, it should be identified and reversed the change that triggered the build to break down. It takes too long for tests to take place. According to DORA research, the experiments will not take longer than a few minutes to run with a maximum duration of around 10 minutes. If the construction takes longer than that, you can increase the testing performance, add additional machine resources to run them in parallel, or split longer running tests into a separate construction using the deployment pipeline model [25]. Not often enough fusion into a trunk. Many companies have automated tests and constructions, but they do not perform a daily trunk fusion. This results in long-term branches which are much more difficult to integrate and lengthy feedback loops for developers.

IV. CONCLUSION

CI guarantees the software is continuously running and the divisions of developers are not substantially different from the trunk. Studies show that CI has a greater deployment rate, more reliable systems, and better-quality applications. The benefit of CI is significant. The main aspects of continuous integration effectively should involve an engagement that stimulates a software build. A sequence of test automation should be activated in a few minutes for each commit. One needs the following to incorporate these elements:

An automated process of construction. The initial phase in CI is to provide an automated script that creates packages for any environment. The latest research supports this claim by helping to emphasize the links between software creation, design, and implementation. Continuous integration in software development projects helps to prevent disconnections and mitigate risk.

REFERENCES

[1] Schaefer, A., Reichenbach, M., and Fey, D., 2012, "Continuous Integration and Automation for DevOps", Lecture Notes in Electrical Engineering, pp. 345-358.

[2] Childs, H., Brugger,E.S, Bonnell,K.S., Meredith,J.S., Miller,M., Whitlock,B.J., Max,N.,2005, "A contract-based system for large data visualization. In: Proceedings of IEEE Visualization, pp 190–198.

[3] Berg, A. M., 2015, "*Jenkins continuous integration cookbook: Over 90 recipes to produce great results from Jenkins using pro-level practices, techniques, and solutions*," Birmingham, UK: Packt Publishing.

[4] Cois, A.,2015, "Continuous Integration in DevOps", Insights.sei.cmu.edu.Available at: https://insights.sei.cmu.edu/devops/2015/01/continuous-integration-in-devops-1.html: 16- Jul- 2020].

[5] Elbaum, S., Rothermel, G., and Penix, J.,2014, "Techniques for improving regression testing in continuous integration development environments. In FSE.

[6] Hüttermann, M.,2012, "*DevOps for developers*," Berkeley, CA: Apress.

[7] Duvall, P. M. (2010). *Continuous integration: Patterns and anti-patterns*. Durham, NC: DZone, Inc.

[8] Hunt A, and Thomas, D., 1999, "The pragmatic programmer: from journeyman to master," Addison-Wesley, Boston.

[9] Kawalerowicz, M., and Berntson, C., 2011, "*Continuous Integration in .NET: Includes index*," Greenwich, Conn: Manning.

[10] Bosch, J., 2014, *Continuous software engineering*, Cham: Springer.

[11] Smart, J. F., 2011, "*Jenkins: The definitive guide*," Sebastopol, CA: O'Reilly Media.

[12] Melymuka, V., 2012, "*TeamCity 7 continuous integration essentials*," Birmingham: Packt Publishing.

[13] Brechner, E., and Waletzky, J.,2015). *Agile project management with Kanban*. Redmond, WA: Microsoft Press.

[14] Vanbrabant, B., and Delaet, T., 2010, "Authorizing and directing configuration updates in contemporary its infrastructures. In: Proceedings of the 3rd ACM workshop on assurable and usable security configuration, SafeConfig '10, New York, NY, USA, ACM, pp 79–82.

[15] Reichenbach, M., Schmidt, M., Pfundt, B., and Fey, D., 2011, "A new virtual hardware laboratory for remote FPGA experiments on real hardware. In: Proceedings of the 2011 international conference on e-Learning, e-Business, enterprise information systems, e-Government, EEE '11.

[16] Lasserre, J.,2009, "*Linear and Integer Programming Vs. Linear Integration and Counting: A Duality Viewpoint*," Dordrecht: Springer. Internet resource.

[17] Gruver, G., Young, M., and Fulghum, P., 2013, "*A practical approach to large-scale Agile development: How HP transformed LaserJet FutureSmart Firmware*, "Upper Saddle River, NJ: Addison-Wesley.

[18] Eckstein, J., & O'Reilly for Higher Education (Firm). (2013). Agile Software Development in the Large: Diving Into the Deep. Addison-Wesley Professional.

[19] Sacks, M. (2012). *Pro website development and operations: Streamlining DevOps for large-scale websites*. New York: Apress.

[20] Taiko, P. F., 2009, "Designing software-intensive systems: Methods and principles, Hershey: Information Science Reference.

[21] Sacks, M., 2012, "*Pro website development and operations: Streamlining DevOps for large-scale websites*," New York: Apress.

[22] Allspaw, J., and Robbins, J.,2010, "Web operations," Beijing [China: O'Reilly.

[23] Bergmann, S., 2011), *Integrating PHP projects with Jenkins*. Sebastopol, Calif: O'Reilly.

[24] Humble, J., & Farley, D. (2011). Continuous delivery. Upper Saddle River, NJ: Addison-Wesley.

[25] Kroll, P., & Kruchten, P. (2003). *The rational unified process made easy: A practitioner's guide to the RUP*. Boston: Addison-Wesley.

[26] Brown, E., 2014, "*Web development with Node and Express*, "Sebastopol, CA: O'Reilly Media.