INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 7, Issue 3 - V7I3-1510)
Available online at: https://www.ijariit.com

A Study on Development Operations for Continuous Integration using Jenkins

Deepika R.
deepikar.sse19@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

Sharadadevi Kaganurmath
sharadadeviks@rvce.edu.in
RV College of Engineering, Bengaluru, Karnataka

ABSTRACT

DevOps is a new way of thinking for software design that intends to break down the barriers between developers and operations team to provide quick delivery and immediate response to changing requirements throughout the software development Process. Each programmer or testing team will be allocated to different project, so it would be critical to test and build the modules to ensure they operate working efficiently and without flaws. One of the important techniques among software developers is Continuous Integration (CI), in which developers integrate their work into a baseline on a regular basis. Automating the process like build and testing is the important way to make Continuous Integration more powerful and quicker. This paper examines the challenges faces during traditional Integration and advantages of better CI approach using Jenkins.

Keywords– *Continuous Integration (CI); DevOps; Jenkins; Software Development; Version Control System*

1. INTRODUCTION

Over the past two centuries, the technology world has dependent on a typical approach to delivering continuous improvements in the software development life cycle, which has its own set of drawbacks and challenges, such as late delivery, prolonged time to market, inconsistent product quality, etc. Software technologies have matched up with the new emerging developments as technological innovations evolve over time, resolving some of these drawbacks.

One of them will be Continuous Integration, which is a game-changing technology that overcomes the complexities of conventional software development. Numerous software technologies have been developed, and some have been revised throughout based on changes in market requirements. So, without automation, deployment will take a long time, since every time people modify a little piece of code, then must re-deploy it. Using Jenkins, users can simplify the integration process in just a fraction of minute through button click soon after the making improvements to the project, which is especially helpful in organizations and universities where almost all systems are interconnected via Network.

Continuous Integration is a development framework wherein programmers periodically commit their works, which will then be built and tested before being approved. One of the important and best Open Source Continuous Integration (CI) platforms is Jenkins which is a Java Based tool, that allows users to build and analyze projects remotely. Jenkins' main job is to perform predefined steps recognized as jobs in response to a trigger. A modification in Version Control System (VCS), such as Git or Gerrit will be tracked every 30 minutes and will work as a trigger to integrate the changes. Jenkins can operate in a master-slave framework, in which a master can run various slaves on separate systems to perform jobs as shown in Fig 1.

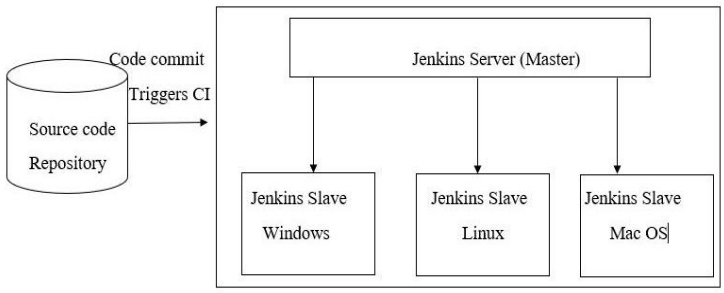

Fig. 1: Jenkins master-slave Architecture

2. BRIEF LITERATURE REVIEW

2.1 Continuous Integration, Delivery and Deployment

Mujtaba Shahin et al. conducted a thorough evaluation of techniques, tools, difficulties, and practises and found a link between continuous integration, delivery, and continuous deployment. These three strategies are software development industry practises that enable firms to release new features and products on a regular and consistent basis. A Systematic Literature Review (SLR) of methodologies, methods, problems, and practises identified in empirical investigations on continuous practises has been offered in this article.

2.2 Automation Using Jenkins: Plugins, Test Design, Test Execution and Reporting

Mrs. Kavitha S N and Arpita R have provided a full description of Jenkins, a popular Continuous Integration tool. Jenkins is a self-contained, open-source Automation Server that may be used to test, create, and deploy applications. The paper focuses on test execution, reporting, and various Jenkins plugins, as well as how to best use the plugins for automation.

2.3 Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible

The relevance of automation utilising modern DevOps tools was discussed by Sriniketan Mysari and Vaibhav Bejgam. They concentrated on constructing and integrating with Jenkins utilising pipeline approach, as well as delivering with the Ansible tool, and provide a rudimentary skeleton of it. Saving resources such as time is a key difficulty for emerging firms, thus automation in integration and deployment using Jenkins ansible saves a lot of time, and it can also be readily updated if there are frequent changes that need to be made in the project. The shh access is straightforward to set up with Ansible, and it can be executed on any Jenkins node.

2.4 ACI (Automated Continuous Integration) using Jenkins: Key for Successful Embedded Software Development

Integration is a critical component of SD for any S/W enterprise, according to Nikita Seth and Rishi Khare. For CI automation, Jenkins is a superior option. They demonstrated how to use Jenkins to develop an automation script for building and running CTS. They looked at a real-world example, such as how software development is done in corporations, and how Jenkins can save developers and integrators valuable time by automating the entire process. Jenkins can also work in a master-slave design, where a master can execute certain jobs on several slaves running on different platforms, according to the report.

2.5 A Prologue of Jenkins with Comparative Scrutiny of Various Software Integration Tools

Various Software Configuration tools have been discussed by Preethi Rai et al. They discuss Jenkins, an open source continuous integration tool that operates in a servlet-like container and is basically a server-oriented structure. Among other Source Control Management (SCM) technologies, it supports Subversion, Mercurial, Perforce, ClearCase, and Rational Team Concert (RTC). The goal of this research is to look at the Jenkins Integration Development Environment (IDE) and compare five software integration solutions to see how useful and effective they are.

3. DISADVANTAGES OF TRADITIONAL CI

- When all the Developers had finished their assigned coding tasks, they would all commit their code at the same time. Build is then reviewed and deployed. A single build was completed after several days, and code commits, and test cycles were infrequent.
- Since the code was designed all at the same time, some developers would have to wait until others finished coding before checking their builds.
- Isolating, detecting, and fixing errors for several commits is a difficult process.
- Since the code building and testing process is completely manual, there are many opportunities for failure.
- A developer must wait for the result from a central repository every time they commit code changes.
- The conventional system lacks a continuous feedback loop, resulting in a short software product life cycle.
- There is a lot of manual quality and assurance testing, which leads to production errors.
- There is a lack of communication between the developers and the operations team.
- Earlier, the industry relied on a method named as Nightly Builds, in which an automated process pulls and builds code that's been added to a centralized database.
- However, discovering and repairing bugs was a real pain because the code installed at night was too big. As a result, rather than concentrating on the main product, the developer was more concerned with addressing these issues.

4. PROPOSED SYSTEM

The proposed approach involves the best practices of DevOps life cycle such as Continuous Integration. The system uses Jenkins as key tool for integrating various code changes done by the Developers. We address a need for Continuous Integration and Test Analysis Automation, as well as the different Jenkins plugins, which is an open source platform. In fact, we consider a master/slave framework, in which a Jenkins platform will simultaneously run jobs on different clients. Jenkins is one of the open source CI servers which is Java Based. Due to its free, open source, and modular nature, it has occupied the top spot in DevOps toolchains for the past few years. It's used to continually create and test software projects, making it easier to integrate changes. It can run projects built with Apache Ant and Maven, as well as shell scripts and Windows batch files. It comes with hundreds of plugins to help you create and test virtually any project.

4.1 Overview of Continuous Integration with Jenkins

The most significant aspect of DevOps is Continuous Integration, which is utilised to connect multiple DevOps processes. The most well-known Continuous Integration tool is Jenkins. It is a Java-based open-source automation platform with installable plugins designed for Continuous Integration. Jenkins is used to continually create and test your software projects, making it easier for developers to integrate changes to the project and for users to get a fresh build. It also enables you to release your software on a

continuous basis by interacting with a variety of testing and deployment technologies. Organizations can use Jenkins to automate the software development process and speed up the process. Jenkins integrates a variety of development life-cycle operations, such as build, document, test, package, stage, deploy, static analysis, and more. Jenkins uses plugins to accomplish Continuous Integration. Plugins allow various DevOps stages to be integrated. If you want to integrate a certain tool, you must first instal the tool's plugins. Git, Maven 2 projects, Amazon EC2, HTML publishers, and so on.

4.2 Workflow of CI with Jenkins

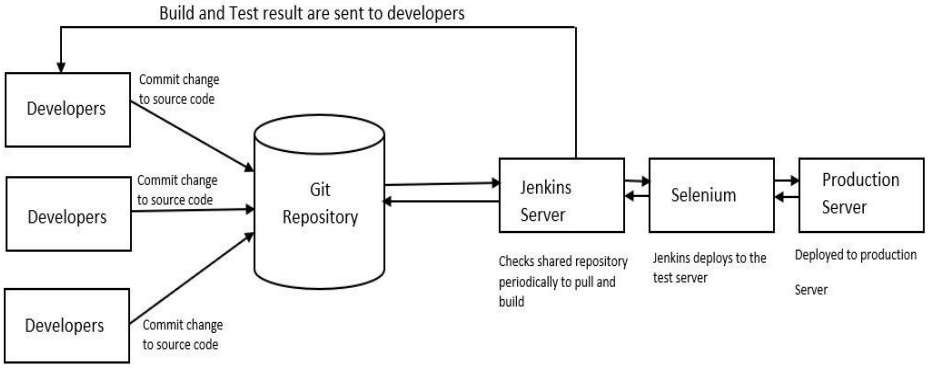

Fig. 2: Flow Diagram of CI using Jenkins

The code is committed to the source code repository by a developer. Meanwhile, the Jenkins server monitors the repository for updates at regular intervals. Jenkins server detects the changes in the source code repository shortly after a commit is made. Jenkins will pull those updates and begin working on a fresh build. The concerned team will be contacted if the build fails. Jenkins delivers the built in the test server if the build is successful. Jenkins generates feedback after testing and then alerts the developers of the build and test results. It will keep checking the source code repository for updates to the source code, and the process will continue.

4.3 Jenkins Plugins

Plugins are the main source of customizing the features of a Jenkins environment to meet the needs of an entity or a single person. Over a thousand plugins may be put on a Jenkins controller to combine various build tools, cloud providers, analytical tools, and more. The Update Center may automatically download plugins and their dependencies. The Jenkins Update Center is a service provided by the Jenkins project that keeps track of open source plugins created and maintained by members of the Jenkins group.

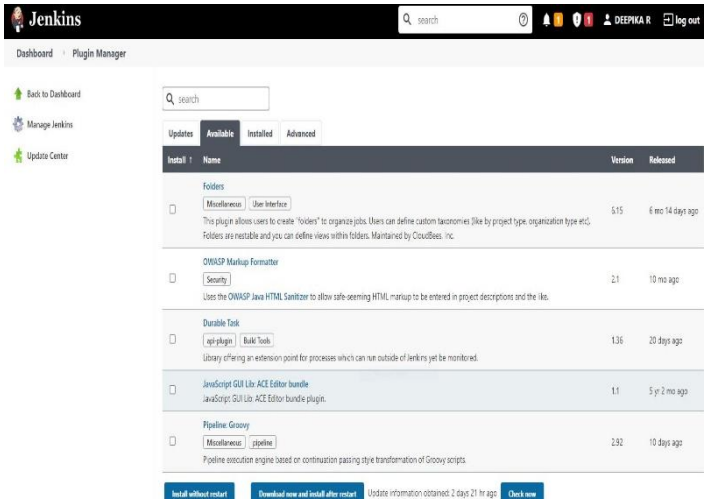

Fig. 3: Jenkins Plugin Manager view

4.3.1 Docker Plugin for Jenkins: The goal of this Docker plugin is to be able to use a Docker host to dynamically setup a Docker container as a Jenkins agent node, conduct a single build on that node, and then take down that node without the build process (or Jenkins job specification) requiring any knowledge of Docker.Jenkins is configured with knowledge of one or more docker hosts (or swarms), one or more "templates" (which explain the labels/tags that this template supplies, the docker image, how to start it, and so on) by the Jenkins administrator. Jenkins may then use docker containers to supply Jenkins (agent) Nodes for running builds.

4.3.2 Jira Jenkins Plugin: It connects Jenkins and Jira to provide a better view of the Software development process. Using this plugin, user can send data automatically from Jenkins to Jira about builds and installations, as well as monitor issues from the Jira side towards Jenkins.

4.3.3 Slack Integration Plugin: Slack is unique plugin it not only provides a chat client; it also includes a network that helps the user to integrate with different applications. Even Jenkins can also be integrated with Slack. Jenkins will send build status updates to a defined channel after you instal the plugin.

4.3.4 Maven Integration: Jenkins supports Maven natively, but it lacks a bit sophisticated when it comes to integration process. Maven plugin is available in different version and offers improved version Maven 2/3 project integration.

4.3.5 Jenkins Pipeline: It's a fantastic addition that was made possible by the Jenkins 2.0 version. It helps you to combine multiple build jobs into a single workflow by scripting your build pipeline. It allows pipeline visualisation much simpler, as well as tracking which sections are running, and they also keep track of pending job which are in queue.

4.3.6 Amazon EC2: Jenkins' are advantageous to find out to many nodes and share the lined work among them is one of its lesser-known features. If your own infrastructure is insufficient, this plugin allows user to use Amazon EC2 infrastructure tools. User can also choose which AMIs to be preferred, give the required labels to them, provide link credentials which is needed, and best test connectivity.

4.3.7 JUnit Plugin: In Jenkins, the most popular types of builds are those that validate the application. It will run JUnit tests by default and provide the results. This plugin is needed if you want to gain a better understanding of the test trends. User can see the available results easily in graphical visualisation, log failures and others with this plugin.

4.3.8 Mailer Plugin: It's just as straightforward as its name suggests. User may use the Mailer plugin to set up email build alerts performance. This plugin also supports many stable protocols like SSL and TSL and offers a number of configuration choices to send a mail with respect to desired situation.

4.3.9 Green balls Plugin: Jenkins has the blue colour traditionally used to indicate good build which gets passed. As a result, blue is used as the colour of performance in all status messages in the GUI. With the Green Balls addon, you can change the colour to green if you prefer.

4.3.10 Git Plugin: For Jenkins Multibranch Pipelines and Jenkins Organization Folders, the git plugin supports a multibranch provider. The multibranch provider in the git plugin is a "basic implementation" that leverages command-line git. When a multibranch solution is available, users should use it for their git provider. REST API calls can be used to improve the Jenkins experience and add new features for multibranch solutions for various git providers. GitHub, Bitbucket, GitLab, Gitea, and Tuleap all provide multibranch implementations**.**

5. ADVANTAGES OF USING JENKINS CI

- Reduced development time – Since each commit is designed and checked, new features can be released to users quicker and with less mistakes.
- Code integration takes less time – Before Jenkins CI, code integration was performed manually, which took a few days. In certain instances, the code might not be in a running state, making it difficult to debug because it has passed through several commits in the repository. Integrating code after each commit means that at the very least the functionality is not disabled.
- Developer teams get faster reviews – When a test fails during a commit, developers receive feedback and can immediately enhance the code. Otherwise, debugging the problem could be challenging because teams wouldn't know which commit caused the error.
- Teams don't have to think about running a manual test for each commit because the workflow is automated. The Jenkins CI pipeline verifies the most recent code and builds it alongside the tests. It can deploy the project in a specific environment if the test is green; otherwise, it can alert the developer by breaking the construct.
- Around 280 tickets have been resolved so far, and the project releases a stable update every three months.
- Jenkins evolves in tandem with technology. Jenkins currently has about 320 plugins in its plugins database. Jenkins becomes even more strong and feature-rich with plugins.
- Jenkins also has cloud-based architecture support, allowing you to use Jenkins on cloud-based platforms.
- Jenkins' popularity stems from the fact that it was built by a developer for developers.

6. CONCLUSION

Continuous integration systems enable team participants to concentrate their efforts on the most important issues, reducing development time and improving quality of the product. The development team may focus more time on software design, and the machine gets rid of the redundant integration job. Testers can be adequately checked with continuous quick feedback information. The Continuous Integrated Delivery System (CIDS) represents a watershed moment in automated service and maintenance. It aids in improving the maturity of software projects, implementing continuous lean process enhancement, promoting software service level improvement, and promoting the high-quality evolution of software systems through operation a.

7. REFERENCES

[1] S. Mysari and V. Bejgam, "Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, 2020, pp. 1-4, doi: 10.1109/ic-ETITE47903.2020.239

[2] W. Yiran, Z. Tongyang and G. Yidong, "Design and implementation of continuous integration scheme based on Jenkins and Ansible," 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, 2018, pp. 245-249, doi: 10.1109/ICAIBD.2018.8396203

[3] V. Armenise, "Continuous Delivery with Jenkins: Jenkins Solutions to Implement Continuous Delivery," 2015 IEEE, pp. 24-27, doi: 10.1109/RELENG.2015.19.

[4] N. Seth and R. Khare, "ACI (automated Continuous Integration) using Jenkins: Key for successful embedded Software development," 2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS), Chandigarh, 2015, pp. 1-6, doi: 10.1109/RAECS.2015.7453279.
[5] P. Rai, Madhurima, S. Dhir, Madhulika and A. Garg, "A prologue of JENKINS with comparative scrutiny of various software integration tools," 2016 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2016, pp. 201-205.
[6] V. Armenise, "Continuous Delivery with Jenkins: Jenkins Solutions to Implement Continuous Delivery," 2015 IEEE/ACM 3rd International Workshop on Release Engineering, Florence, 2015, pp. 24-27
[7] V. Debroy and S. Miller, "Overcoming Challenges With Continuous Integration and Deployment Pipelines: An Experience Report From a Small Company," in IEEE Software, vol. 37, no. 3, pp. 21-29, May-June 2020
[8] K. Gallaba, "Improving the Robustness and Efficiency of Continuous Integration and Deployment," 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA, 2019, pp. 619-623
[9] M. Soni, "End to end automation on cloud with build pipeline: the case for DevOps in insurance industry continuous integration continuous testing and continuous delivery", *2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pp. 85-89, 2015, November.
[10] R. Pietrantuono, A. Bertolino, G. De Angelis, B. Miranda and S. Russo, "Towards continuous software reliability testing in DevOps", *2019 May*, vol 5, issue 3 pp. 21-27.

