IJIEMR Transactions, online available on 2nd Dec 2022. Link :http://www.ijiemr.org/downloads.php?vol=Volume-11&issue=Issue 12

DOI: 10.48047/IJIEMR/V11/ISSUE 12/01

Title A Review on Continuous Integration of Jenkins with Industrial Use cases

Volume 11, ISSUE 12, Pages: 1-6

Paper Authors
Nitesh Kaushik,Abhinav Khandelwal,Shiv Kumar Agarwal, Rishi Kumar Sharma

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

A Review on Continuous Integration of Jenkins with Industrial Use cases

Nitesh Kaushik[1*],Abhinav Khandelwal[2],Shiv Kumar Agarwal[3,]Rishi Kumar Sharma[4]
[1,3,4]Associate Professor, Faculty of Computer Science & Engineering, Poornima University, Jaipur, Rajasthan, India
[2]UG Student, Faculty of Computer Science & Engineering, Poornima University, Jaipur, Rajasthan, India

ABSTRACT

An open source automation server is called Jenkins. Continuous integration and continuous delivery are made possible by it by automating the software development processes of developing, testing, and deploying. It is a server-based program that runs on Apache Tomcat or another servlet container. It can run arbitrary shell scripts, Windows batch operations, and version control programs including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, Clear Case, and RTC. It also supports Apache Ant, Apache Maven, and set-based projects. Continuous integration and continuous delivery may significantly lessen the issue of finding and fixing bugs slowly. A company can produce several releases that adhere to the deadline as long as the time needed to detect, and repair issues is decreased. For the continuous integration process, several software solutions, such as Jenkins, Bitbucket, and TeamCity, have been created.

Keywords: Jenkins, Automation Server, GUI, Apache, CVS, CI, Pipelines.

1.INTRODUCTION

Everything from workstations on corporate intranets to powerful servers connected to the public internet use Jenkins. Jenkins provides many configuration choices for activating, modifying, or turning off different security elements to safely serve this vast range of security and threat profiles. To ensure that Jenkins is safe, several of the security options are enabled by default after the interactive setup wizard is completed. Others depend on use cases enabled by unique Jenkins instances and necessitate setting up and making trade-offs specific to the Continuous Integration (CI) is a process in software development that involves integrating the changes made by the developers who contribute to the source code at the time the code is submitted.
With the help of automated building and testing, the integration is confirmed. Upon a successful test, the Deployment testing done of builds. [1,2]

1.1 USE CASES

1.) Jenkins lowers the Effort of repeated coding
Jenkins makes it possible to convert command-line code into GUI button clicks, which minimizes the work required for repetitive writing. To accomplish this, the script might be packaged as a Jenkins task. Jenkins jobs can take user input or be parameterized to be customized. Thousands of lines of code can be saved as a result.

2.) Integration of Individual Jobs
Jenkins tasks are frequently straightforward tools. They succeed in a few, modest objectives. Jenkins provides a pipeline plugin that makes it possible to combine many tasks. Pipelining provides this benefit, and Linux users are best suited to take advantage of it. Combinations may be parallel or sequential.

3.) Synchronization with Slack
A sizable team communicates via a centralized platform. Slack is one of these most well-liked services. Jenkins may connect to Slack so that details about triggered actions, their times, users' names, results, etc. may be communicated with others.

4.) Effortless Auditing
Jenkins can perform database auditing thanks to this plugin. The person who started the build, the node where it is executed, and any build parameters (if any) can all be recorded in the database. [3]
XML files are already used to store Jenkins activities.

The rest of the paper is organized as follows: Section 2 describe about the Continuous Integration, Section 3 describes about the research findings which include resources and procedures, pipelines, Section 4 includes Jenkins- CI workflow orchestration, Jenkins- CI project and plugins, Jenkins-CI cell profiler, and Overview of pipeline automation, Section 5 includes Conclusion and Future Scope and lastly Section 6 tells about the references.

2. CONTINUOUS INTEGRATION(CI)

Programmers are required to often integrate their work into a main repository as part of a process known as continuous integration. Instead of creating new features from scratch without any quality assurance, each change is compared to the central repository to identify issues. Each developer makes daily contributions to a shared mainline, and every contribution kick starts an automated build and test process. A faulty build or test can be identified and fixed in a couple of minutes without endangering the entire structure, process, or project. It is possible to isolate issues in this way to fix them more rapidly and create products of higher quality. [4]

3. RESEARCH FINDINGS

The Jenkins Integration Development Environment (IDE) is highlighted, and five distinct software integration solutions are reviewed and compared to ascertain their usefulness. According to the investigation, Jenkins can help with the quick and easy correction of severe flaws in contrast to other technologies. [3]

Jenkins has developed from a Continuous Integration platform to a Continuous Delivery platform. It has embraced the automated build, release, and delivery processes used in modern design. The challenges that Jenkins tracking capabilities must overcome are listed. When moving from CI to CD, there are a few problems that must be resolved, including versioning artifacts that must be constantly shippable and monitoring the environment where artifacts are generated. Although many attempts have been made to overcome these issues and many solutions have been proposed, a more thorough investigation is required.
Software patches are integrated and distributed to consumers using Jenkins. The advantage of using the Jenkins tool for software development in business undertakings is presented using a real-world example. Developers might gain time at work by automating the entire process. Jenkins is implemented using a master/slave architecture, as seen in Figure 2, where the master node is the Jenkins server and the slaves are the Jenkins clients. We go into great length on the value and use of the numerous plug-ins that are available to aid in development. [5]

The automation scripts that have been established can expand in the future.
In a continuous integration system, source code and build script maintenance is done automatically. The first strategy entails conducting an empirical examination of software build failures and build repair trends. A method for using build scripts to automatically remedy build problems has been created based on the findings of the empirical investigation. It is suggested to extend this repair procedure to include both the build script and the source code. It is advised that user research be done to quantify the automatic fixes and offer a comparison between the fixes produced by the suggested approach and actual repairs. [4]

When an integration is finished, the code is built and tested. It could be difficult for the developers to comprehend some of the problems that were discovered during testing. Testing large code blocks can occasionally be time-consuming. To mitigate this, the developers might resort to making significant changes to the code, which might ultimately slow down the integration process. A novel testing approach is proposed in which testing is performed on a small fraction of the updated code rather than the entire piece of code, i.e., the test is concentrated on a single area of the entire code. A "micro-pipeline" concept is presented.

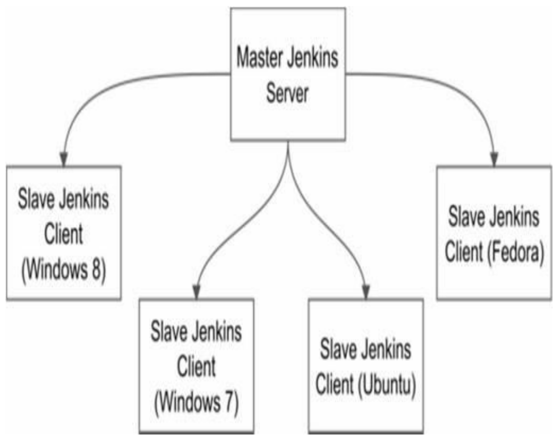

Master Jenkins Server
Slave Jenkins Client (Windows 8)
Slave Jenkins Client (Fedora)
Slave Jenkins Client (Windows 7)
Slave Jenkins Client (Ubuntu)

Fig 1. Jenkins Master/ Slave Configuration
In this figure it tells about the Jenkins Master or slave Configuration, which includes Ubuntu,fedora, windows 7, windows 8.

3.1 RESOURCES AND PROCEDURES

1. Image Capture
With a 10 objective, cell images were taken using the IN Cell 2000 (GE Healthcare, Buckinghamshire, UK) for small interfering RNA (siRNA) screening. FITC was exposed for 0.7 seconds, compared to 0.1 seconds for DAPI. Flat-field correction was used on both channels. Pictures were taken using the DAPI (80 MS exposure) and FITC (1000 MS exposure) channels with a binning value of 1 1 for the phosphoprotein identification test using an IN Cell 2000.

2. Tools for configuring software are gaining popularity every day. In this work, we introduce Jenkins, an open source continuous integration solution that is mostly server-oriented and operates in a container like a servlet (like, Apache Tomcat). Various Source Control Management (SCM) tools, such as Subversion, Mercurial, Perforce, Clear Case, and Rational Team Concert, are supported (RTC). Jenkins design feature and applications and compare five software integration. [9]

3.2. PIPELINE

A Jenkins pipeline is made up of several tasks, processes, or events that are linked to one another in a certain sequence. It is, in other words, a group of plugins that enable it simple to use and integrate continuous delivery pipelines. Extensible automation aids a pipeline in the creation of both straightforward and complex delivery pipelines using domain-specific language, or DSL. [6]

The Jenkins pipeline automates many CI/CD pipeline operations, making them extremely predictable, efficient, repeatable, and high-quality. Jenkins oversees attaching tasks with a specific format to the pipeline. It stands for the pipeline's continuous delivery, DevOps life cycle management, and SDLC activity integration. The diagram below shows the Jenkins pipeline.

This comprises Build, Deploy, Test, and Release activities that are continuously integrated and delivered. It is done because these tasks depend on one another and because the pipeline has a specific shape. To reduce the cost, time, and number of iterations without compromising the quality, this entails continuous automation.

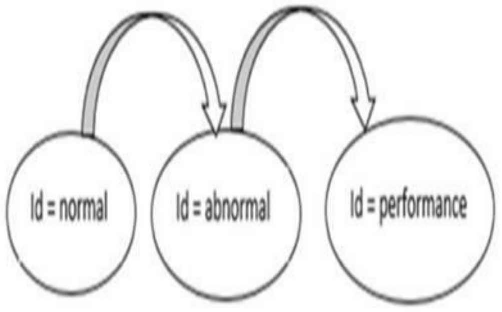


Fig 2. Micro-Pipeline for division unit

4.JENKINS-CI WORKFLOW ORCHESTRATION

After image capture, scientific data- and image-processing operations are integrated and coordinated via Jenkins-CI. Figure 1 summarizes the key elements of a typical Jenkins-CI system, supplemented with compute and data architectural components of our implementation. [7]

High-capacity networked data shares mounted on different operating systems make up the data architecture we chose to support HPC image processing, which is further discussed under Results. The mapping of Jenkins language to more widely recognized business process and workflow principles is provided by Supplemental Table 6.1.

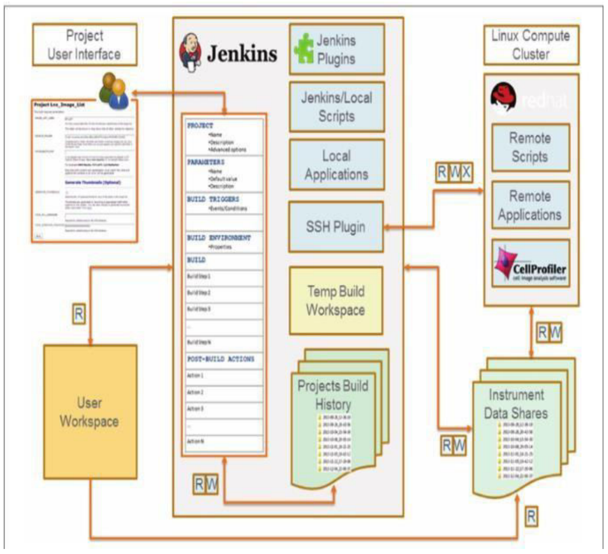
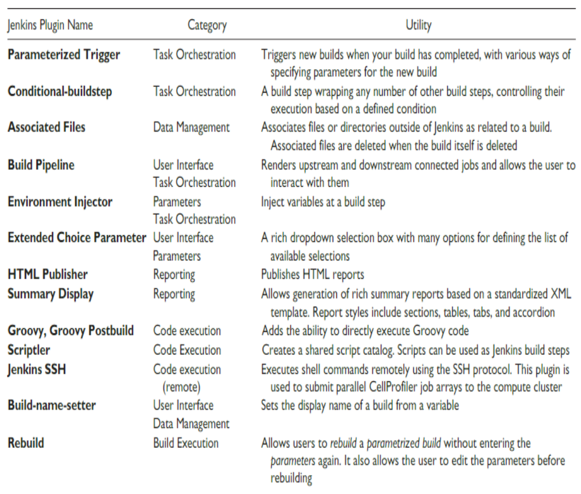


Fig.3 Architecture of Jenkins-CI considered as a scientific data-processing platform. A typical Jenkins-CI installation integrates computational resources and local remote and data and makes them accessible to end users via a standard web portal.

4.1 JENKINS CI-PROJECTS AND PLUGINS

Each project is an illustration of the Jenkins-CI "free-style, parameterized build" project type. Every Jenkins-CI project has a specific sequence of build phases. Each construction step has functionality provided by a corresponding Jenkins plugin (such as data copying, archiving, running commands, and writing custom scripts). The required custom code is created using Groovy scripts, which may then be executed using the Groovy plugin. To build multistage pipelines, we connect numerous jobs using the Parameterized Trigger plugin. Cell Profiler is performed through multistage pipelines that integrate output data and keep track of a cluster run. We may interact with specific stages and see workflow pipelines using the Build Pipeline plugin. Table 1 lists the Jenkins plugin that we make use of. [8]



Jenkins Plugin Name	Category	Utility
Parameterized Trigger	Task Orchestration	Triggers new builds when your build has completed, with various ways of specifying parameters for the new build
Conditional-buildstep	Task Orchestration	A build step wrapping any number of other build steps, controlling their execution based on a defined condition
Associated Files	Data Management	Associates files or directories outside of Jenkins as related to a build. Associated files are deleted when the build itself is deleted
Build Pipeline	User Interface Task Orchestration	Renders upstream and downstream connected jobs and allows the user to interact with them
Environment Injector	Parameters Task Orchestration	Inject variables at a build step
Extended Choice Parameter	User Interface Parameters	A rich dropdown selection box with many options for defining the list of available selections
HTML Publisher	Reporting	Publishes HTML reports
Summary Display	Reporting	Allows generation of rich summary reports based on a standardized XML template. Report styles include sections, tables, tabs, and accordion
Groovy, Groovy Postbuild	Code execution	Adds the ability to directly execute Groovy code
Scriptler	Code Execution	Creates a shared script catalog. Scripts can be used as Jenkins build steps
Jenkins SSH	Code execution (remote)	Executes shell commands remotely using the SSH protocol. This plugin is used to submit parallel CellProfiler job arrays to the compute cluster
Build-name-setter	User Interface Data Management	Sets the display name of a build from a variable
Rebuild	Build Execution	Allows users to *rebuild* a *parametrized build* without entering the *parameters* again. It also allows the user to edit the parameters before rebuilding

4.2 JENKINS CI – CELL PROFILER

By incorporating Cell Profiler into the Jenkins workflow, the Jenkins-CI Cell Profiler project parallelizes and launches Cell Profiler on the Red Hat Linux cluster. Utilizing the Distributed Resource Management of the Univa Grid Engine, the Cell Profiler activity is parallelized (DRM). Jenkins uses a Groovy build step to produce the required grid engine task array script. Using a sequence of shell commands on the distant Linux cluster, we launch the task array, supply the required Cell Profiler command line options, and set up the environment for Cell Profiler to execute in "headless mode" (that is, from the command line without a user interface). Groups of 12 picture sets make up the divisions in image lists. One grid engine processes each batch of images. Cell Profiler task selects the range of photos in the image list to be processed using the first and last image set command line options. [10]

4.3 OVERVIEW OF PIPELINE AUTOMATION

1.It is said that the Pipeline is a "Black Box."

When creating a highly autonomous pipeline, it is essential to keep an eye on what is currently running and where attributes are being delivered. The remaining quality

It is necessary to have a dark box with no view of what is happening.

2. A company cannot utilize the Pipeline again.

It is a bottom-up approach that supports organizational teams since it is all about doing things. Organizations are therefore urged to develop solutions that work for themselves to advance quickly. However, enterprises must reproduce tried-and-true solutions because they cannot afford to experiment any farther than at the corporate level. [11]

3. No changes could be made to the Pipeline

The fact that planned pipelines disintegrate during unexpected changes while still performing better on successful routes is a significant issue. Scripted pipelines are prone to failure, whether they are represented in code or a flow diagram, and they necessitate extensive redesign if the pipeline's inputs or outputs change.

4. The Pipeline will not take obedience or safety into account.

Typically, the team members construct the apps' distribution pipeline.

Their job is to write software into development, test it, distribute it to various environments, and then track it to generate functionality. But finally, a couple other important factors, particularly safety and obedience, must be properly considered in any institutions.

5. The Pipeline disregards procedures pertaining to the commercial sector

Unavoidably, there is a risk of excessive automation. The fact that automation must continue implies that not everything should be automated, though. There have always been pipeline components that require user intervention, such as the decision to scale back following quality concerns, the consent to go live, or a pipeline jump that cannot simply be automated.

6. Offering end-to-end pipeline accessibility in the software development sequence to everyone is one measure to address the issues mentioned above.[13]

a. Giving teams throughout the enterprise a framework that makes it simple for them to utilize pipelines.

b. Enabling a model-driven approach by standardizing updates and implementations that can easily adapt to changes in fields and applications.

c. Integration of security and management across the whole pipeline.

d. Conveniently integrating human and automated processes.

The overview of the CI stages is shown in figure 3 below. Version control, ongoing software integration, delivery, and deployment are the four essential processes. These four acts will be monitored continuously

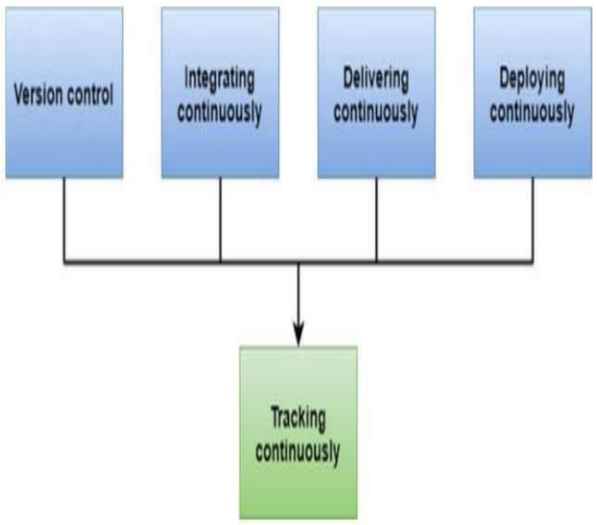


Fig 4. Overview of Continuous Integration

5. CONCLUSION AND FUTURE SCOPE

Since CI/CD shortens the time needed to find and fix errors in a large source code base, it is essential to use it when building software. This paper discusses the challenges of using Jenkins to implement CI/CD, such as the inability to quickly find and fix bugs, the advantages of Jenkins over other development tools, and ways to improve the current approach, such as by using micro-pipelines to reduce testing time and having methods to fix build failure. Jenkins's plug-ins and the problems that must be resolved while transitioning from CI to CD are also highlighted.

Accelerated release cycles, cloud native development, and increased stability are what Jenkins will focus on in the future. Jenkins' extensibility, community, and general purpose have all contributed significantly to its growth over the past ten years. But now that they are more apparent, it still has several problems and shortcomings.

6. REFERENCES

[1] "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," M. Shahin, M. Ali Babar, and L. Zhu, IEEE Access, vol. 5, pp. 3909–3943, 2017, doi: 10.1109/ACCESS.2017.2685629.

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

[3] P. Rai, Madhu Rima, S. Dhar, Mahilika, and A. Garg, "A prologue of JENKINS with comparative examination of several software integration tools," in InaCom, 2015, 2nd International Conference on Computing for Sustainable Global Development, pp. 201-205.

[4] V. Arsenide, "Continuous Delivery with Jenkins: Jenkins Solutions to Implement Continuous Delivery," 2015 IEEE/ACM 3rd International Workshop on Release Engineering, pages 24-27, doe: 10.1109/RELENG.2015.19

"ACI (automated Continuous Integration) using Jenkins: Key for effective embedded Software development," N. Seth and R. Khare, 5 RAECS 2015, 2nd International Conference on Recent Advances in Engineering & Computational Sciences, 2015, pp. 1-6, doi: 10.1109/RAECS.2015.7453279.

[6] F. Hassan, "Tackling Build Failures in Continuous Integration," 34th IEEE/ACM International Conference on Automated Software Engineering, 2019, pp. 1242–1245, doi: 10.1109/ASE.2019.00150.

[7] "Test Automation Framework for Implementing Continuous Integration," 2009 Sixth International Conference on Information

Technology: New Generations, pp. 784–788, doi: 10.1109/ITNG.2009.260

[8].Faheem Ullah, Adam Johannes Raft, Mojtaba Shahin, Mansooreh Zahedi and Muhammad Ali Babar, "Security Support in Continuous Deployment Pipeline," Proceedings of 12th International Conference on Evaluation of Novel Approaches to Software Engineering, 2017.

[9] Valentina Armenise, "Continuous Delivery with Jenkins," IEEE/ACM 3rd International Workshop on Release Engineering, pp. 24-27, 2015.

[10] Zebula Sampedro, Aaron Holt and Thomas Hauser, "Continuous Integration and Delivery for HPC," Practice and Experience in Advanced Research Computing, pp. 22-26, 2018.

[11] S.A.I.B.S. Arachchi and Indika Perera, "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management," 2018.

[12] Wang yiran, Zhang tongyang and Guo yidong, "Design and Implementation of Continuous Integration scheme based on Jenkins and Ansible," pp. 245-249, International conference on Artificial Intelligence and Big Data, 2018.

[13] Nikita Seth and Rishi Khare, "ACI (Automated Continuous Integration) using Jenkins: Key for Successful Embedded Software Development," Proceedings of RAECS UIET Punjab University Chandigarh, 2015.