JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Exploring the Effectiveness of Jenkins CI/CD Pipelines: A Comprehensive Survey

Prof. R. P. Arbat
Assistant Professor
College of Engineering and Technology, Akola

Abstract –Continuous Integration (CI) and Continuous Delivery (CD) have become integral practices in modern software development, enabling teams to deliver code changes swiftly and reliably. Jenkins, an open-source automation server, plays a pivotal role in facilitating CI/CD pipelines, offering a robust framework for automating build, test, and deployment processes. This study delves into the effectiveness of Jenkins CI/CD pipelines, which are pivotal in modern software development for ensuring rapid and reliable code deployment. Jenkins, an open-source automation server, serves as the backbone for orchestrating CI/CD processes, enabling automation of various stages from code integration to deployment. Through a comprehensive review of literature and practical insights, this research assesses the performance of Jenkins pipelines, focusing on metrics such as efficiency, reliability, scalability, and flexibility . By comparing Jenkins with alternative CI/CD platforms, this study elucidates the unique advantages and contributions of Jenkins in enhancing software delivery workflows. The findings underscore Jenkins' crucial role in empowering development teams to streamline their processes, improve productivity, and deliver high-quality software with confidence.

Keywords:- Continuous Integration, Continuous Delivery, pipeline, Jenkins.

Introduction

CI and CD refer to continuous integration and continuous delivery or continuous deployment. Simply put, CI is a contemporary software development methodology wherein frequent and dependable incremental code changes are made. Through CI, automated build and test procedures are initiated to ensure the reliability of code changes being merged into the repository. Subsequently, these changes are swiftly and seamlessly delivered as part of the CD process. In software development, the CI/CD pipeline denotes the automation facilitating the swift and dependable delivery of incremental code changes from developers' environments to production.

Jenkins serves as an open-source automation server, empowering organizations to streamline the software development cycle through automation. It oversees and orchestrates various aspects of software delivery across the entire lifecycle, encompassing build, documentation, testing, packaging, staging, deployment, static code analysis, and more. Jenkins can be configured to monitor code changes from platforms like GitHub, Bitbucket, or GitLab, and automatically initiate builds using tools like Maven and Gradle. Leveraging container technologies such as Docker and Kubernetes, Jenkins facilitates test execution and enables actions such as rolling back or rolling forward in production environments.

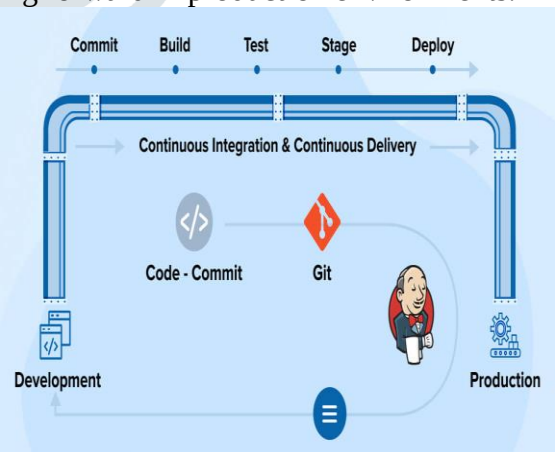

Fig. Ci/Cd Pipeline.

The purpose of this paper is to highlight the key performance of Jenkins for Continuous Integration and continuous Development of a software.

Why Choose Pipeline?

Jenkins serves as an automation engine, supporting various automation patterns. The Pipeline feature enhances Jenkins by providing a robust set of automation tools, accommodating use cases ranging from basic continuous integration to extensive continuous delivery pipelines. By defining a sequence of interconnected tasks, users can leverage Pipeline's numerous features:

Code-Centric Approach: Pipelines are defined in code and typically stored in version control, enabling teams to modify, review, and refine their delivery processes collaboratively.

Resilience: Pipelines can persist through both planned and unexpected restarts of the Jenkins controller, ensuring continuity in the automation process.

Human Interaction: Pipelines can be configured to pause and await human input or approval before proceeding with the next steps, facilitating manual interventions when necessary.

Flexibility: Pipelines support complex CD requirements, allowing for branching, merging, looping, and concurrent execution of tasks to accommodate diverse deployment scenarios.

Customization: The Pipeline plugin offers extensive customization options through its Domain Specific Language (DSL) and seamless integration with other Jenkins plugins.

I. Literature Survey

Project development typically involves three main phases: development, testing, and production. In the software industry, the traditional Waterfall methodology was initially prevalent. For instance, in a project with a 12-month timeline, these phases could be divided accordingly: development spanning six months, testing spanning three months, and production spanning three months. During the development phase, which spans six months, developers write the code for the project. Subsequently, the developed code undergoes testing to identify any errors or bugs. If errors are detected during testing, the testing team notifies the development team, prompting them to debug the code. Once the code is modified, it is resent to the testing team for further evaluation.

Mysari and Bejgam recently conducted research on CI/CD Pipeline Automation using Jenkins and Ansible for continuous integration and continuous development, respectively. In their work, they utilized YAML language in Ansible to create a default website in an IIS server. The study identified several key advantages, including a reduction in code review time and the efficient use of CI/CD through Jenkins for building web applications.

Additionally, Sampedro et al. presented an approach integrating Singularity containers with Jenkins and Puppet to facilitate the adoption of CI/CD practices in High-Performance Computing (HPC) workflows. This integration aimed to enhance the delivery of high-quality and reliable software. In this setup, Jenkins agents run in Docker containers managed by Docker Compose configurations, with delivery jobs facilitated through the Jenkins SSH plugin composed configuration. For delivery jobs Jenkins SSH plugin is used.

Mandale and Dhoble presented Jenkins CI/CD pipeline with Github integration. In this framework The CI/CD pipeline encompasses shared repositories hosted on GitHub, where distinct branches cater to various environments. These branches include a feature branch for adding new functionalities, a develop branch for pre-production environment testing, and a master branch for the live production environment utilized by consumers. Amazon EC2 instances are utilized to automate the execution of pipelines upon repository changes, ensuring servers are automatically updated. The GitHub repository architecture is modeled after the Git Flow architecture, albeit with some modifications. CI/CD Jenkins pipelines are triggered by designated GitHub repositories in response to different actions such as branch creation, pull requests, merges, and releases. Through these pipelines, developers can seamlessly incorporate changes or introduce new features to an application, confident that the tested features will be automatically deployed to pre-production and production servers. The key advantages found in this work are- Increase in productivity, team can make smaller code changes, team can isolate faulty code changes, CI/CD pipeline helps the team to reduce the faster mean time to resolution also reliability of web application improves due to smaller size.

In their study, Kavya N and Smitha P showcase the capabilities of various tools in building a single-container website, illustrating the functionality of each technology. They have developed a website utilizing AWS services and tools such as Jenkins, Git, Maven, Terraform, Docker, and Kubernetes for automation purposes. Amazon EC2 serves as the computing service for the project. Jenkins is employed for automation and CI/CD deployment processes. Specifically, a Jenkins job is configured to monitor GitHub for any new code changes and to integrate them into the system.

II. Architecture of Jenkins CI/CD Pipeline:

This diagram and the steps that follow describe the CI/CD pipeline architecture:

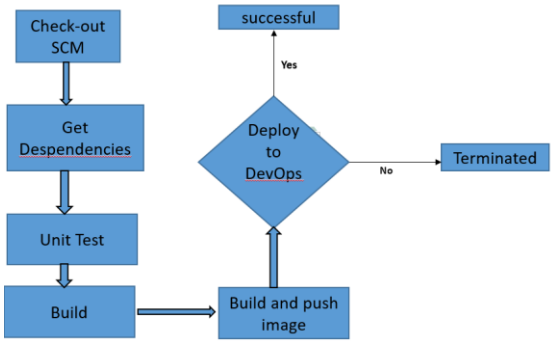

Fig. Architectural flowchart

- Check your repository.
- Compile the binary.
- Conduct a multi-architecture container image build, including AMD64 and s390x architectures, create a manifest, and add it.
- Execute pre-integration testing.
- Upload the multi-architecture image to the DockerHub container registry.
- Deploy an OpenShift application from the image on both AMD64 and s390x architectures, and expose it to the internet.

Jenkins employs a master-worker architecture to handle distributed builds, with each component serving a distinct purpose:

Jenkins master oversees the scheduling of build jobs and delegates the actual execution to the workers. It also monitors the status of the workers and aggregates the build results on the web dashboard.
Jenkins worker, also referred to as a build agent, is a Java executable deployed on a remote machine. It listens for commands from the Jenkins master and executes the assigned build jobs. The number of workers can be scaled as needed, with the ability to add or remove them dynamically. This
allows for automatic workload distribution, alleviating the load on the master Jenkins server."

III. Brief view on Advantages using Jenkins CI/CD Pipeline

Free and Open Source: Developers and DevOps teams often prefer solutions that are open source and free of charge.
Plugins and integrations: A key benefit of Jenkins is its extensive collection of plugins accessible for the platform. These plugins are open for development by any user, catering to a wide range of needs and preferences.
Strong Community Backing: Jenkins was initially established in 2011 and underwent several iterations prior to that, accumulating a rich operational history as a CI/CD solution. Its open-source nature empowers creators, community contributors, and users to actively engage in enhancing the tool's functionality, upkeep, and future direction. The Jenkins community boasts a user base exceeding one million individuals
Integration with other CI/CD platforms: While Jenkins provides unique advantages in software development, it's important to recognize that it's not the sole option for implementing a CI/CD pipeline, nor the exclusive CI solution within an organization. For instance, Jenkins can be seamlessly integrated with another platform utilized by a software team for continuous delivery. Alternatively, teams may opt to employ a different tool for the CI and build stages of their pipeline, leveraging Jenkins solely for the building and storage of application artifacts. Numerous plugins available for Jenkins facilitate integration with various CI/CD platforms. Examples include plugins for Azure DevOps and Azure DevOps Server (formerly known as Team Foundation Server), as well as the GitLab plugin.
Flexibility: Developers prioritize writing tests to promptly identify errors within their code, thereby avoiding prolonged debugging sessions during large-scale integrations. By swiftly detecting and addressing issues, the software remains in a deployable state, ensuring it can be released safely at any given moment. Automation plays a significant role in streamlining integration tasks, resulting in fewer integration-related challenges. This not only saves time but also reduces project costs over time.

IV. Conclusion

In summary, Continuous Integration (CI) and Continuous Delivery (CD) have revolutionized software development, enabling rapid and reliable code deployment. Jenkins plays a central role in this process, offering a versatile platform for automation. Its Pipeline feature facilitates seamless integration and delivery through a code-centric approach, resilience, flexibility, and extensive customization options. Existing literature highlights Jenkins' efficiency in various domains, from web applications to High-Performance Computing (HPC) workflows. The architectural overview emphasizes Jenkins' distributed setup, ensuring scalability and efficient workload distribution. With its open-source nature, extensive plugin ecosystem, and strong community support, Jenkins remains a top choice for developers and DevOps teams, streamlining software delivery and ensuring readiness for deployment.

V. References

[1] Sriniketan Mysari and Vaibhav Bejgam "Continuous Integration And Continuous Deployment PipelineAutomationUsing Jenkins Ansible" 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE).

[2] Rismanda Tyas Kusumadewi, Ronald Adrian "Performance Analysis of Devops Practice Implementation Of CI/CD Using Jenkins", Journal of Computer Science and Information Technology), 2023.

[3] Zebula Sampedro, Aaron Holt, Thomas Hauser" Continuous Integration and Delivery for HPC Using Singularity and Jenkins" PEARC '18, July 22–26, 2018, Pittsburgh, PA, USA.

[4] Hritik Mandale, Prashant Dhobale, Parag Ganorkar, Deveshri Daware "JENKINS CICD PIPELINE WITH GITHUB INTEGRATION", International Research Journal of Modernization in Engineering Technology and Science, Volume:05/Issue:07/July-2023.

[5] Arpita S.K, Amrathesh, Dr. Govinda Raju "A review on Continuous Integration, Delivery and Deployment using Jenkins", Journal of University of Shanghai for Science and Technology, Volume 23, Issue 6, June – 2021.

[6] Miss. Kshitija Patil, Miss. Sayali Kapadnis, Mr. Ravindra Waghmare, Mr. Harshal Thakare, Prof. Mr. Rahul Raut "Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins", International Journal of Scientific Research in Engineering and Management (IJSREM) Volume: 06 Issue: 11, November – 2022.

[7] Kavya N, Smitha P "Deploying and Setting up Ci/Cd Pipeline for Web Development Project on Aws Using Jenkins", International Journal of Advances in Engineering and Management (IJAEM) Volume 4, Issue 6 June 2022.

[8] Niranjan DR, Mohana "Jenkins Pipelines: A Novel Approach to Machine Learning Operations", Proceedings of the International Conference on Edge Computing and Applications (ICECAA 2022).

[9] *www.google.com*