International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 04 | April - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

AWS TECHNOLOGY: The Ultimate CI/CD Devops Pipeline Technology with the Help of Jenkins.

Chinmay B. Patole
Department of Computer Engineering
Atharva College of Engineering Mumbai, India
patolechinmay-cmpn@atharvacoe.ac.in

Ayush Sankhe
Department of Computer Engineering
Atharva College of Engineering Mumbai, India
sankheayush-cmpn@atharvacoe.ac.in

Sohan Patil
Department of Computer Engineering Atharva College of Engineering Mumbai, India
patilsohan-cmpn@atharvacoe.ac.in

Nihar R. Patil
Department of Computer Engineering
Atharva College of Engineering Mumbai, India
patilnihar-cmpn@atharvacoe.ac.in

Dr. Suvarna Pansambal
Department of Computer Engineering
Atharva College of Engineering Mumbai, India
suvarnashirke@atharvacoe.ac.in

***ABSTRACT* — A CI/CD pipeline is an automated framework that coordinating, tests, and conveys code changes ceaselessly, allowing for quicker and more solid updates to applications. A Continuous Integration/ContinuousDeployment (CI/CD) pipeline utilizing Jenkins, is popular open-source mechanization server. The pipeline will automate the build, test, and arrangement of computer program applications, guaranteeing speedier time to- market, progressed quality, and diminished manual exertion. Automated testing and reporting are moreover executed, enabling high-quality program discharges and recognizing abandons early in the advancement cycle. The pipeline too manages different situations, such as development, staging, and production, ensuring smooth and efficient deployment of applications.**

Keywords:
Jenkins, CI/CD, Pipeline, Automation, Integration, AWS (Amazon Web Services).

I.INTRODUCTION

In today's fast-paced software development landscape, the ability to quickly and reliably deliver high-quality software applications is crucial for business success. To achieve this, organizations need to adopt efficient and automated development practices that minimize errors, reduce manual effort, and accelerate time-to-market. This is where Continuous Integration and Continuous Deployment (CI/CD) pipelines come into play. The Jenkins CI/CD Pipeline project automates software development, testing, and deployment ensuring faster and more reliable delivery of high-quality software products. By leveraging the power of Jenkins, a leading automation server, this project aims to streamline the software development lifecycle, improve collaboration among development teams, and reduce the risk of errors and downtime. The CI/CD pipeline will integrate with version control systems, automate testing and reporting, manage multiple environments, and provide robust rollback and recovery mechanisms. By implementing this pipeline, our organization can improve the speed, quality, and reliability of software releases, ultimately driving business success and customer satisfaction , improve the speed, quality, and reliability of software releases, ultimately driving business success.

II.LITERATURE SURVEY

Sriniketan Mysari and Vaibhav Bejgam[1] explains that Automation is what every company needs to handle large projects where two or more people work together. CI/CD using Jenkins ansible is an optimized way to build and host any web applications. They focused on building and integrating with Jenkins with the pipeline methodology and deploying with Ansible and gives you its basic skeleton. Integrating with Jenkins is an easy task because

it is open source and has many plug-in options . Ansible is an open-source configuration management tool , uses the YAML configuration language for ease of use.

Mandale and Dhoble [2] proposed a framework where the CI/CD pipeline encompasses shared repositories hosted on GitHub, where distinct branches cater to various environments. These branches consist of a master branch for the live production environment that user access, a feature branch for adding new features, and a develop branch for testing in pre-production environments. Amazon EC2 instances are utilized to automate the execution of pipelines upon repository changes, ensuring servers are automatically updated. The GitHub repository architecture is modelled after the Git Flow architecture, albeit with some modifications.

F. Zampetti, et al introduced Continuous Integration and Continuous Delivery is the new method of software integration [3] with the existing repo. Developers typically upload their modifications to a repository like Bit Bucket, SVN, VSS, AWS Code Commit, Git Hub, etc. Given that the workplace culture is Currently distributed in nature, every commit made by several developers is dispersed throughout the repository. Therefore, in order to push the code into a release/master branch of the repository, dependencies are reduced.

Artur Cepuc and Robert Botez [4]. proposed that Nowadays, cloud computing has become a solution for most businesses. This has led to the introduction of DevOps techniques, in which developers work closely with network engineers to ensure rapid and reliable deployment of their applications. This document presents the entire automated pipeline, starting from detecting changes in the source code of a Java-based web application, creating new resources in Kubernetes cluster to host this new version, and finally deploying a containerized application in AWS. Jenkins is used in the continuous integration phase of the solution, which adheres to DevOps best practices.

S.A.I.B.S. Arachchi and Indika Perera explains Agile processes with continuous integration and continuous delivery (CICD) have increased the efficiency of projects [5]. Every sprint delivery in agile style adds new functionality to the system, and even if a delivery is well-developed, it could fail because problems with performance. Due to the delivery timeline, a common solution in such situations is to switch to system scaling. But how much should the system scale? System scaling requires the current state of the system benchmark and the expected state of the system. Production benchmarking is a critical task because it interrupts a live system. The new version should go through a stress test to measure the expected health of the system.

A. Ankit, K. Nimala and M. Jadhav[6] have proposed various frameworks and architectures for implementing CI/CD pipelines using cloud computing. Study presented a framework for automating the build, test, and deployment process of software applications using cloud-based services. The study builds upon these existing works and proposes a novel approach to creating a CI/CD pipeline using cloud computing. The authors' proposed framework leverages cloud-based services to automate the build, test, and deployment process of software applications, improving efficiency, scalability, and quality. The study’s findings are consistent with existing research

S. R. Dileepkumar and J. Mathew[7] investigates the integration of Jenkins-based Continuous Integration and Continuous Deployment (CI/CD) into software development processes. They claim that Jenkins- based CI/CD pipelines can deliver speed, quality, and value. Jenkins and CI/CD literature and Jenkins based CI/CD pipeline literature and a software engineering case study using Jenkins based CI/CD pipelines was review as a methodology. The case study showed that the use of Jenkins based CI/CD pipelines enhances the speed, quality, and efficiency of software engineering processes. The authors of this paper were not the first to conduct this research, which implies that Jenkins-based CI/CD pipelines improve software development productivity.

III.PROPOSED SYSTEM

Since a single DevOps pipeline doesn't fit all, gaps need to be filled depending on the company's technology stack. A DevOps engineer, irrespective of the organization's budget, skill, etc., should have knowledge about everything ranging from development to operations, DevOps tool chains, infrastructure, systems administration, and even programming.
In addition, no two companies can have the same set of things that may affect the process. There are only three main components that focus on this area:

○
ontinuous Integration/Continuous Delivery

● **Continuous integration** refers to the practice of making frequent code changes and committing the code to a shared source repository. The integration of a new code modification into the existing code base facilitates quick detection and simpler resolution of disputes arising from code changes done by different programmers.

●
ontinuous delivery is where the “main” or “trunk” branch of source code for an application is maintained perpetually in a releasable state.

● **Continuous deployment** entails placing new software versions in production environments without human activities because they are so well tested and operated on.

○
ontinuous Feedback

●
ontinuous testing is a substantial factor of every DevOps project pipeline and is one of the primaries enablers of continuous feedback. In a DevOps, from development to testing to deployment, modifications are made continuously, which results in both quicker releases and better-quality products.

●
ontinuous monitoring is another important component of continuous feedback. A Devops approach entails using continuous monitoring in the staging, testing, and even development environment.

○
ontinuous Operation

C ● **Continuous operations** is a kind of new and less common term, and definitions vary. One way to understand it is as "uninterrupted operation.".

● Continuous operations can also be thought of as continuous alerting.

● This is the idea that if there are any performance issues with the application or infrastructure, engineering staff is alerted and on call. In the majority of C situations, continuous monitoring and alerting go hand in hand.

IV.METHODOLOGY

Developing a plan for DevOps deployment is essential to guaranteeing that DevOps principles are successfully adopted within a company. The roadmap should outline clear objectives, C milestones, and timelines for implementing DevOps processes and tools. By aligning the DevOps implementation roadmap with business C goals and technological constraints, organizations can create a structured approach to implementing DevOps and realizing its benefits across development and operations teams.

1. Initiate the DevOps Initiative

➢ The journey starts with high-level support, and the DevOps project is crucially launched by a CIO or IT director.

➢ In this step, the initiative must be in line with the organization's larger IT and business objectives, the required funding must be obtained, and a program manager must be assigned to supervise the creation and implementation of the strategy.

2. Define DevOps Objectives and Goals

➢ A company's success in using DevOps depends on setting clear goals and objectives. By establishing clear goals for software delivery, quality, and operational effectiveness, teams may coordinate their efforts to produce measurable results.

➢ Establishing measurable KPIs helps track progress, identify bottlenecks, and drive continuous improvement in DevOps practices. By defining clear goals, organizations can ensure that their DevOps initiatives are focused, impactful, and aligned with business objectives

3. Develop a Comprehensive DevOps Strategy

➢The key to a successful DevOps transformation is a strong strategy. Establishing a common vision that promotes cooperation across development, operations, testing, and other divisions is necessary to achieve this. Infrastructure as Code (IaC) integration for effective infrastructure provisioning, build-to-deployment procedures, and speeding up the software development cycle without sacrificing quality should be the major goals of the strategy.

4. Establish DevOps Team Structure

➢DevOps teams are usually made up of people with a variety of talents, such as developers, operations engineers, DevOps engineers (also called cloud, platform, or site reliability engineers), and quality assurance specialists, who collaborate to achieve shared objectives.

➢Within the DevOps team, clearly defined roles and duties encourage openness and responsibility, allowing team members to comprehend their contributions to the organization's success as a whole. By fostering a cooperative and encouraging team atmosphere, businesses can use DevOps techniques to promote innovation & long-term.

5. Implement Containerization

➢ Using solutions like Docker to containerize your apps improves software independence and dependability, making deployment smooth and consistent across many environments. Additionally, by separating modifications to particular microservices within containers, this step streamlines operations management.

6. Integrate Infrastructure With CI/CD Tools

➢ Integrating infrastructure automation tools (e.g., Kubernetes, Ansible, Chef) with CI/CD tools (e.g., Jenkins, Bamboo, GoCD) allows for proper containerized application administration. This combination ensures efficient configuration management, fault tolerance, continuous monitoring, and streamlined software updates and deployments.

7. Automate Testing and Align QA with development

➢ While not all testing can or should be

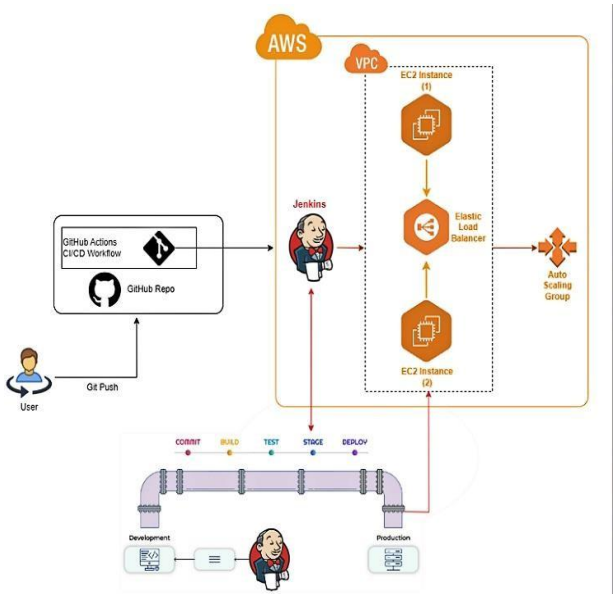
automated, determining the optimal balance is critical. Automate where it makes sense to shorten delivery times and boost test coverage. Simultaneously, promote tight collaboration between the QA and development teams to facilitate early detection and resolution of issues, hence improving product stability and quality

8. Monitor Application Performance

➢ Continuous monitoring of application performance is critical for identifying, prioritizing, and addressing issues. Use application performance monitoring tools to acquire insights into software behavior, user experience concerns, and system bottlenecks, allowing for rapid troubleshooting and optimization.

V.DETAILS OF DESIGN, WORKING & PROCESSES

- Architecture



This architecture shows a complete CI/CD pipeline integrating GitHub Actions and Jenkins. Developers push code to GitHub, which triggers the workflow. Jenkins manages the build, test, and deployment processes. The application is deployed onto AWS EC2 instances, placed behind an Elastic Load Balancer for traffic distribution. Auto Scaling Groups ensure high availability and scalability based on load. This setup streamlines development to production delivery in a reliable and automated manner.

VI.FUTURE SCOPE

DevOps is evolving towards closer integration between software development and operations, emphasizing streamlined processes and enhanced team collaboration.

1. Streamlining DevOps with ML/AI

The trajectory of DevOps is increasingly influenced by emerging technologies, driving significant improvements in software development and delivery. Artificial intelligence (AI), machine learning (ML), and serverless computing are transforming the environment by enabling process automation, resource efficiency, and speedier delivery.

2. DevOps ensures secure & robust applications

Incorporating security and compliance controls early in development aligns with the DevSecOps concept and provides secure applications. The combination of cloud technologies, microservices, and containerization facilitates the development of flexible and scalable solutions, indicating an impressive move toward more dynamic, secure, and efficient approaches to software development.

3. FinOps for Cloud Cost Optimization

Integrating FinOps into the DevOps framework is a strategic step toward optimizing cloud spending and improving financial responsibility in cloud investments. FinOps, or Cloud Financial Management, refers to a set of techniques that strive

to provide financial transparency in the cloud spending model, allowing teams to make more informed decisions about resource allocation and utilization.

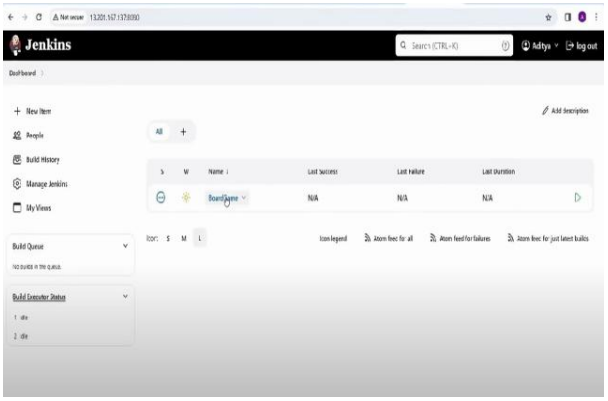
VII.RESULT ANALYSIS

1. Deployment Efficiency and Performance

Jenkins automates software deployment by streamlining CI/CD processes. The deployment efficiency is measured in terms of build success rate, deployment time, and resource utilization. The average deployment time using Jenkins is reduced significantly compared to manual deployment, ensuring faster release cycles. The build success depends on the integration of automation testing and proper pipeline configuration.

2. Error Rate and Debugging Effectiveness

Jenkins provides real-time logs and debugging features that help in identifying and resolving build failures. A lower error rate is observed when proper build triggers, testing automation, and rollback strategies are implemented. Integration with tools like octopus deploy helps in code quality analysis, reducing vulnerabilities.



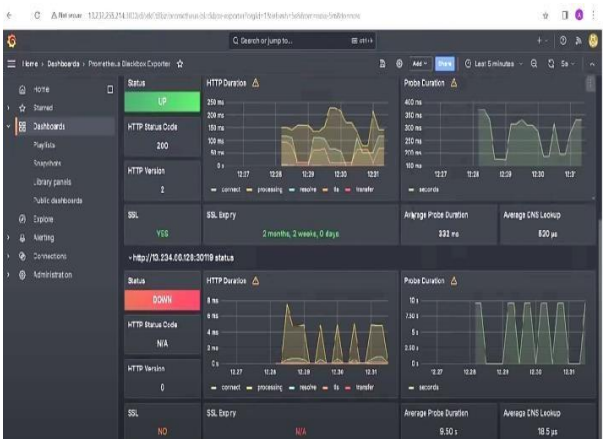
3. Comparison with other Deployment Tools

Jenkins is compared with other tools like TeamCity and Octopus Deploy base on the key performance metrices:

Flexibility: Jenkins offers better flexibility due to its open-source nature and vast plugin eco-system.

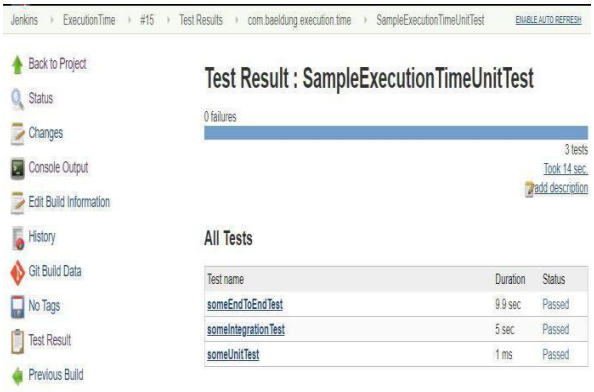
Scalability: While Jenkins scale well with distributed builds, Octopus Deploy provides better release orchestration.

Ease of Use: GUI-based tools like TeamCity may have more intuitive interface compared to Jenkins configuration- heavy setup.



4. Pipeline Optimization and Automation Impact

By implementing multi-stage pipelines, organizations can improve deployment speed and reliability. Parallel processing reduces the overall time for CI/CD execution. Automated rollback mechanisms prevent downtime in case of deployment failure.



5. Key Findings

Jenkins significantly improves deployment speed and automation compared to manual processes. Integrating Jenkins with other DevOps tools enhances error detection and recovery mechanisms. The effectiveness of Jenkins deployment depends on proper pipeline design, resource management, and automation strategies.

VIII.CONCLUSION

To make the cloud transition seamless, efficient, and successful, technology businesses should adopt DevOps concepts and procedures. These concepts are built into AWS and constitute the foundation for many of its services, particularly those related to deployment and monitoring. Begin by specifying your infrastructure as code using the services AWS CloudFormation or AWS CDK. Next, define the way in which your applications are going to use continuous deployment with the help of services like AWS Code Build, AWS Code Deploy, AWS Code Pipeline, and AWS Code Commit. Use containers like AWS Elastic Beanstalk, Amazon ECS, or Amazon Elastic Kubernetes Cluster (Amazon EKS) for your application. To ease the configuration of typical architectures, use AWS Ops Works. Using these services also makes it simple to incorporate other critical services like Auto Scaling and Elastic Load Balancing.

Finally, implement a DevOps monitoring technique, such as Amazon CloudWatch, as well as strong security standards like IAM. With AWS as your partner, your DevOps principles will increase agility in your company and IT department while also accelerating your cloud journey.

IX.REFERENCES

A. Cepuc, R. B. (2020). "Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins, Ansible and Kubernetes. Networking in Education and Research (RoEduNet) (pp. 1-6). Romania: RoEduNet..

Ankit, K. N. (2024). Creation of Continuous Integration Continuous Deployment Pipeline using Cloud. 5th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (pp. 337-341). Tirunelveli, India: ICICV.

Arachchi, S. A. (2018). "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management. Moratuwa Engineering Research Conference (MERCon), (pp. 156- 161). Moratuwa, Sri Lanka.

Hritik Mandale, P. D. (2023). JENKINS CICD PIPELINE WITH GITHUB INTEGRATION. International Research Journal of Modernization in Engineering Technology and Science.

Mysari, S. (2020). Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible. International Conference on Emerging Trends in IT Engineering, 1-4.

Shevchuk, R. (2023). Software for Improve the Security of Kubernetes-based CI/CD Pipeline. Advanced Computer Information Technologies (pp. 420-425). Wroclaw, Poland : (ACIT).

Zampetti, S. G. (2021). "CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study.

International Conference on Software Maintenance and Evaluation (ICSME) (pp. 471-482). IEEE

R. Shevchuk, M. Karpinski, M. Kasianchuk, I. Yakymenko, A. Melnyk and R. Tykhyi, "Software for Improve the Security of Kubernetes-based CI/CD Pipeline," 2023 13th International Conference on Advanced Computer Information Technologies (ACIT), Wrocław, Poland, 2023, pp. 420-425, doi: 10.1109/ACIT58437.2023.10275654

S. R. Dileepkumar and J. Mathew, "Transforming Software Development: Achieving Rapid Delivery, Quality, and Efficiency with Jenkins-Based CI/CD Pipelines," 2023 Annual International Conference on Emerging Research Areas: International Conference on Intelligent Systems (AICERA/ICIS), Kanjirapally, S. R. Vernekar. (2024). Hotel Reservation App Integrated with Docker and Jenkins. Emerging Technologies in Computer Science for Interdisciplinary Applications (pp. 1-6). Bengaluru: (ICETCS) .

India,2023, pp. 1- 6,doi: 10.1109/AICERA/ICIS59538.2023.1042025.