International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins

Prof. Rahul M Raut[1], Kshitija Patil[2], Sayali Kapadnis[3], Ravindra waghmare[4], Harshal Thakare[5]

[1]Professor, Department of Information Technology Sandip Institute of Technology & Research Centre, Nashik, Maharashtra, India-422213.
[2,3,4,5]Student, Department of Information Technology Sandip Institute of Technology & Research Centre, Nashik, Maharashtra, India-422213.

Abstract:
This project focuses on streamlining the software development process by integrating various tools and automating key tasks. Developers contribute code changes to a Git repository hosted on GitHub. These changes trigger Jenkins, a continuous integration and delivery tool, to build the code using Maven, a build automation tool. Maven compiles the code and executes unit tests to ensure its correctness. To access code quality, security vulnerabilities, and bugs, Sonarqube, a static code analysis tool, is employed. As part of the automation process, the built code is packaged into a Docker image. This image is then automatically pushed to Dockerhub, a container registry, with a new version identifier. Additionally, a shell script is utilized to update the manifest deployment file, which specifies the version and configuration details for deployment, with the newly created version. This ensures that the deployment process is seamlessly synchronized with the latest code changes. By integrating these tools and automating the necessary steps, this project enhances the efficiency and reliability of the software development pipeline. It facilitates rapid code testing, quality analysis, and version management, thereby improving the overall development workflow.

Keywords: - DevOps; Cloud; CI/CD; Docker; Jenkins; Maven; Sonarqube; Shellscript; AWS.

1. Introduction

Every company wants to reduce time to market for their products in order to improve customer service for their employees. So, we chose a DevOps approach and used the right technology to launch our application. DevOps can also reduce the cost of your server environment (Amazon Web Services (AWS)) by reducing the load on your servers. This reduces server response time and automates actions. Example: Add resources to a virtual machine. DevOps allows the company to get to market faster.
The primary goal of our approach is to automate the development and deployment process to increase organizational productivity, reduce outages and speed deployment. Deployment automation is a

software delivery method that enables organizations to release new features faster and more frequently. Manual deployment is a straightforward technique. This process is subject to human error. Humans cannot copy as accurately as automated systems, so they run the risk of making many errors. Errors in pre-production processes lead to errors in production, and manual processes have a higher rate of these errors. As such, he uses GitHub Actions as his CI/CD platform to automate application deployments.

2. Related Work / Literature Review

1]Author: Sai Priya Sinde, Bhavika Thakkalapally, Meghamala Ramidi The motivation behind this exploration was to research the way in which programmers expect to use GitHub actions to robotize work processes and the effect of activity reception on pull demands. They accumulated and broke down information from 3,190 dynamic GitHub repositories and found that small set of these repositories utilized GitHub actions, as well as 708 extraordinary predefined actions being utilized in work processes. They additionally assembled and dissected GitHub actions related issues, finding that most of the remarks were positive. By and large, the discoveries show that GitHub's actions were generally welcomed by designers. This article examines the many methodologies and innovations that can be utilized to fabricate a fruitful CI/CD pipeline. They dealt with various apparatuses as well as the issues they confronted while carrying out ceaseless improvement processes in programming 4 advancement using DevOps. This paper presents the thought of on-request benefits, which alludes to the utilization of cloud assets on request and the capacity to scale assets as per request, as well as a survey of safety research in the field of cloud security The approach we used to deploy an application is by using GitHub actions which minimize the number of tools used. Using GitHub actions as a CI/CD pipeline with the AWS EKS cluster, the Flow of deployment is continuous and errorless. The time taken for the deployment of an application with Jenkins as a CI/CD pipeline is 1 min 43 sec approximately. But the time taken for our approach to deploy an application is around 25 sec to 45 sec.[1]

2] Author: Artur Cepuc, Robert Botez ,Ovidiu Craciun Cloud computing has evolved into one of the most ubiquitous concepts in the IT industry. Its success lies in on- demand services rather than deploying a whole infrastructure which would bring additional costs like purchasing and maintaining the equipment, paying the employees and so on. The National Institute of Standards and Technology (NIST) classifies whether or not a service is a cloud service according to the following characteristics broad network access, rapid elasticity, measured service, on-demand self-service and resource pooling. An automated CI/CD pipeline for deploying a Java-based web application on AWS is presented in this paper. While the DevOps best practices were followed, using Ansible in the CD process added something new. We were able to improve overall scalability and send commands directly to the Kubernetes cluster because of this. The results of the experiments demonstrated that the proposed solution is fast, scalable, and reliable, with no downtime. As a result, in just 37.6 seconds, any change to the application's source code is automatically detected and sets off a whole series of six different technologies. The system rolls back the most recent stable version whenever a Jenkins job fails to deploy a new version of the application.[2

3. Motivation

The motivation behind this project is to build a pipeline that drives software development through a code build, test, and deployment path, also known as CI/CD. By automating the process, the goal is to

minimize human error and maintain a consistent process for releasing software. Tools included in the pipeline include code compilation, unit testing, code analysis and security For containerized environments, this pipeline also includes packaging the code into a container image and deploying it on the public cloud.

4. Objective

CI/CD is all approximately automating the software improvement system in order that modifications can be made speedy and easily with out compiling and testing code manually. there are many advantages of using CI/CD, but right here are the vital ones:
1. expanded efficiency: CI/CD allows developers to recognition on writing code as opposed to annoying approximately compiling it or trying out it manually. It approach that they could get greater paintings performed in less time.
2. advanced satisfactory: computerized checking out manner that errors are located and stuck speedy before they have got a risk to motive issues inside the live device.
3. decreased fees: by automating repetitive tasks, groups can save cash on labor costs. similarly, CI/CD can help lessen software program insects, that could keep even more money in the long run.
4. improved collaboration: CI/CD fosters collaboration between developers, testers, and operations teams. through working collectively, they can clear up troubles extra quick and efficaciously.
5. quicker delivery: computerized trying out and deployment mean that new features can be delivered to customers faster. it could be a extensive competitive gain, specially in rapid-shifting industries.

5. Problem Statement

one of the venture necessities is to routinely push the built code image to DockerHub, ensuring that each new version has its own distinct tag. To acquire this, a shell script is employed to replace the deployment manifest record robotically. The script modifies the report to consist of the newly created model, ensuring that the appropriate image is deployed while the software is deployed.
The problem handy includes automating the construct, checking out, code analysis, image pushing, and occur updating strategies, enabling developers to devote code adjustments with self assurance, understanding that the whole pipeline will be executed seamlessly. Additional requirements
1. Setup to be done using AWS, Jenkins.
2. Jenkins should be on the same server as the application being deployed to
Tools
1. Jenkins
2. Git/Bitbucket
3. AWS EC2
4. Maven
5. Sonarqube
6. Trivy
7. Shellscript
8. Docker
9. Putty

6. Proposed Approach /Methodology

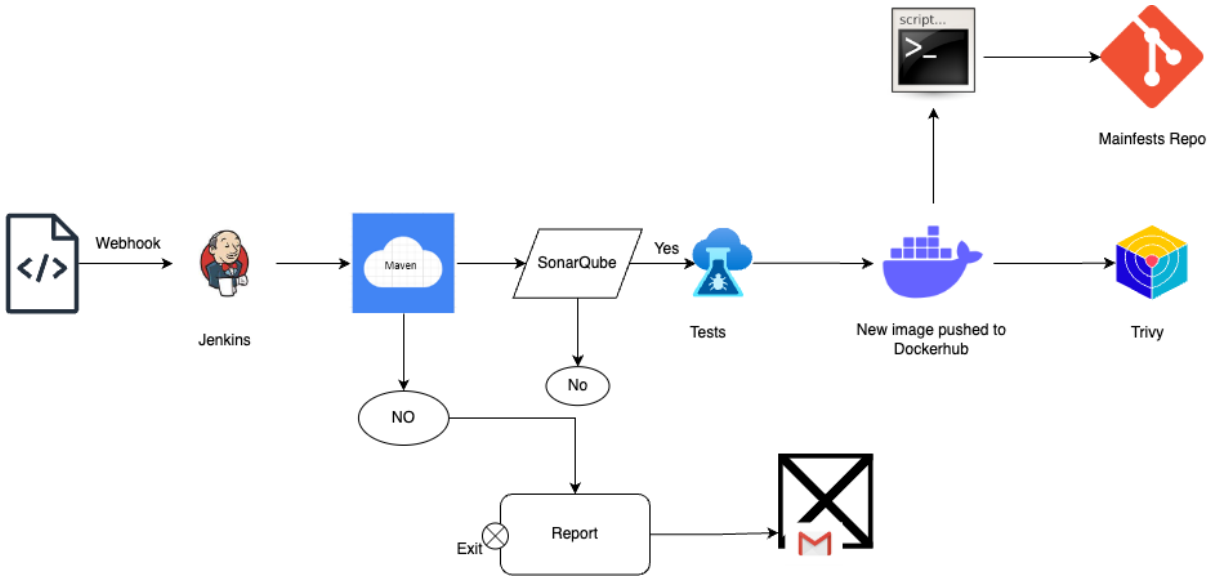

Fig.1.1Proposed Approach

7. Overview Of Project Modules

1. The undertaking follows a streamlined development workflow. developers devote code adjustments to a Git repository on GitHub. Jenkins, configured to use Maven, robotically builds the code and executes unit tests. Sonarqube is employed to carry out comprehensive static code evaluation, detecting code first-class issues, protection vulnerabilities, and insects.
2. deploy the vital Jenkins plugins: Git plugin Maven, Integration plugin, Pipeline plugin, Kubernetes continuous set up plugin
3. Create a new Jenkins pipeline: In Jenkins, create a brand new pipeline task and configure it with the Git repository URL for the Java utility.
upload a Jenkinsfile to the Git repository to outline the pipeline stages.
4. outline the pipeline ranges:
stage 1: take a look at out the supply code from Git.
degree 2: build the Java software the usage of Maven.
stage 3: Run unit exams the use of JUnit and Mockito.
stage 4: Run SonarQube analysis to check the code-best.
degree 5: package deal the utility right into a JAR documents. level 6: replace the manifests repo with deployment document with newer model of photograph the use of ShellScript.
degree 7: scan the Docker photograph that is driven correctly directly to the dockerhub the use of Trivy.
5. Configure Jenkins pipeline tiers: stage 1: Use the Git plugin to check out the supply code from the Git repository. level 2: Use the Maven Integration plugin to build the Java software. stage three: Use the JUnit and Mockito plugins to run unit exams stage 4: Use the SonarQube plugin to analyze the code pleasant of the Java application. level 5: Use the Maven Integration plugin to bundle the application right into a JAR document.

8. Result

The result of the challenge are as follows:

Streamlined Code dedicate technique: builders can easily commit code adjustments to the Git repository hosted on GitHub. This permits for collaborative development and version manipulate, making sure that the codebase stays up to date.

automatic build and checking out: Jenkins, the non-stop integration device, is brought on every time code modifications are pushed to the repository. Jenkins makes use of Maven to build the code and execute unit checks. This automation saves effort and time for the development group, ensuring that the code is constructed constantly and tested very well.

Code fine and safety evaluation: SonarQube is hired to perform static code evaluation. It identifies any code excellent issues, security vulnerabilities, and bugs present within the codebase. by reading the code routinely, the group can pick out and rectify potential troubles early within the improvement procedure.

advanced Code first-class and protection: through the usage of SonarQube, the assignment aims to enhance normal code fine and safety. The static code evaluation helps perceive and address code smells, layout troubles, and ability vulnerabilities. by way of addressing those issues proactively, the crew can reduce technical debt, decorate code maintainability, and decrease protection dangers.

automated photo Pushing: After the code is constructed and tested correctly, the ensuing image is robotically driven to DockerHub. every new edition of the photo is tagged as it should be, permitting easy monitoring and deployment of particular variations. This automation ensures that the state-of-the-art code changes are simply to be had for deployment.

manifest Deployment record replace: A shell script is used to robotically replace the manifest deployment document with the newly created version. This script removes the want for manual intervention and guarantees that the right photograph version is referenced inside the deployment system. The automation simplifies the deployment workflow and reduces the danger of human error.

universal, the challenge result intention to decorate the development process with the aid of automating numerous ranges, ensuring code best, security, and efficiency. the automatic pipeline lets in builders to awareness on coding whilst taking advantage of streamlined build, checking out, evaluation, photo pushing, and deployment strategies.

9.1-Result Screenshot

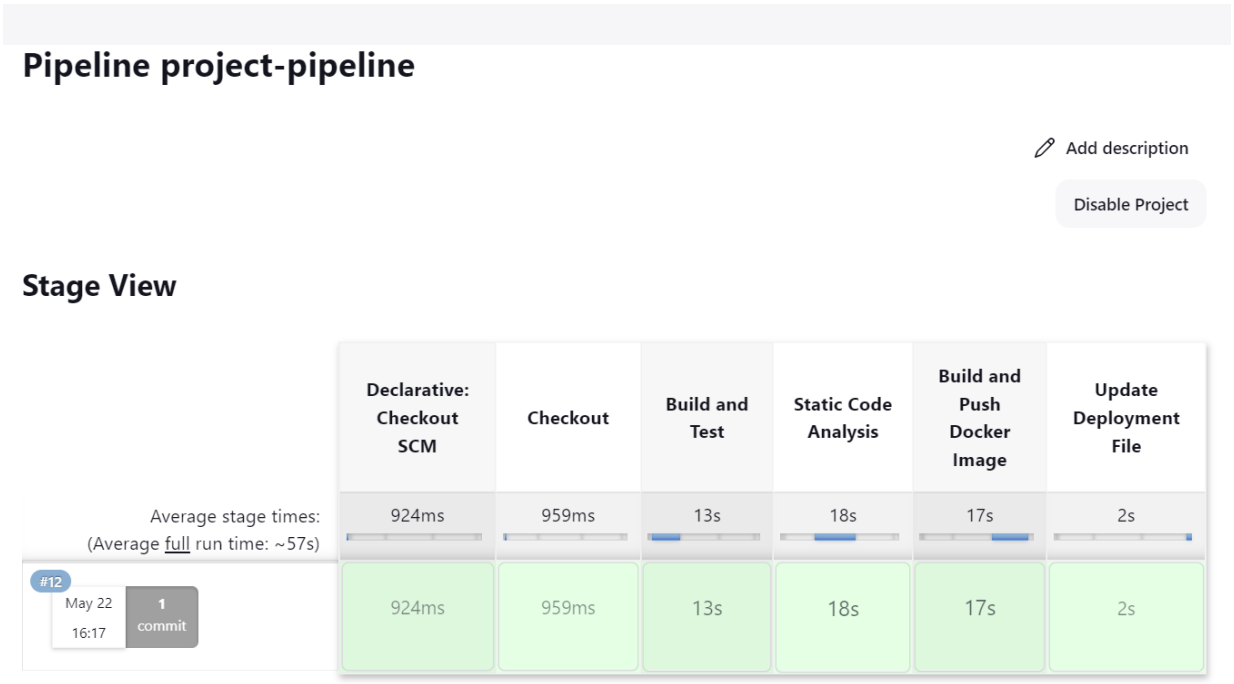

Figure 1.6:Result Sc.

10. Conclusion

This proposed project successfully addresses the development workflow by providing an automated pipeline for code integration, build, testing, code analysis, and deployment processes. Developers are able to commit code changes to the Git repository hosted on GitHub, triggering Jenkins to initiate the build process using Maven. Maven efficiently builds the code and runs unit tests, ensuring the integrity of the codebase.

The inclusion of SonarQube in the pipeline adds an additional layer of quality assurance by performing static code analysis. This analysis identifies and highlights any code quality issues, security vulnerabilities, and bugs, enabling the development team to address these concerns proactively.

To streamline the deployment process, the project implements an automatic image pushing mechanism. After a successful build, the build image is automatically pushed to DockerHub with a new version tag. This ensures that each version of the codebase is stored separately and can be retrieved as needed.

Furthermore, the project utilizes a shell script to automatically update the manifest deployment file with the newly created version. This script eliminates the manual effort required to update the deployment file, reducing the chances of errors and ensuring accurate version tracking during deployment.

Overall, this integrated pipeline enhances the development process by promoting code quality, security, and efficiency. Developers can confidently commit code changes, knowing that the pipeline will handle the build, testing, analysis, image pushing, and manifest updating seamlessly. This automation contributes to faster development cycles, improved collaboration, and a more reliable deployment process.

11. References

1. Sai Priya Sinde, Bhavika Thakkalapally, Meghamala Ramidi, Sowmya Veeramalla "Continuous Integration and Deployment Automation in AWS Cloud Infrastructure" 30 June 2022.
2. DOI: https://doi.org/10.22214/ijraset.2022.44106
3. Artur Cepuc, Robert Botez, Ovidiu Craciun, Iustin-Alexandru Ivanciu and Virgil Dobrota, "Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins, Ansible and Kubernetes", 19th RoEduNet Conference: Networking in Education and Research, 2020. DOI:10.1109/ROEDUNET51892.2020.9324857.
4. Arachchi, S. A. I. B. S., Perera, I., "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management," 2018 Moratuwa Engineering Research Conference (MERCon) DOI:10.1109/mercon.2018.8421965.
5. Sriniketan Mysari; Vaibhav Bejgam "Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible" 27 April 2020. DOI:10.1109/icETITE47903.2020.239
6. Noor Fathima.F;Vani H.Y "Building, Deploying and Validating a Home Location Register (HLR) using Jenkins under the Docker and Container Environment" November 02,2020. DOI: 10.1109/ICOSEC49089.2020.9215458
7. Malathi. S1, Ganeshan. M2, "Building and Deploying a Static Application using Jenkins and Docker in AWS" 4, June 2020. https://www.ijtsrd.com/papers/ijtsrd30835.pdf