Continuous Integration and Continuous Deployment with Jenkins in C++ Software Development

Suhas T M[1], Sowmya Nag K[2]

[1]*Student, Dept of Electronics and Communication, R.V. College of Engineering, Bangalore, INDIA*

[2]*Professor, Dept of Electronics and Communication, R.V. College of Engineering, Bangalore, INDIA*

[1]*suhastm.ec17@rvce.edu.in*
[2]*sowmyanagk@rvce.edu.in*

Abstract: *Continuous Integration is a practice in software program development process where software program builders combine code into a shared repository frequently, more than one instances through the day. Jenkins is a continuous integration tool which assist developer and testers by using automating the entire test, on the way to reduce their work with the aid of tracking the development at each and every stage in software development, each integration push is then tested by means of automated build and test cases, and an easy way to make CI quicker and accelerate CI procedure is to automate the testing of recent build. In this paper a real scenario is taken into consideration, how the software program trying out is performed in corporate sectors and how Jenkins can save developers/testers important valuable hours by automating the whole software development system.*

Keywords: **Jenkins, Continuous Integration, deployment, C++, Automated Testing.**

1. INTRODUCTION

Automation tools enable developers to easily automate the entire process of testing, building and deployment in software development. This differs from manual testing where a human being is responsible for single-handedly testing the functionality of the software in the way a user would. Automation is well-suited for large projects where changes are made more frequently and building those projects and testing the project and finally deployment of the project to recent updates.

In software development, developers work as a team with a shared repository into which new continuous integration features or developed versions are incorporated on a regular basis. , each push can then be verified using automated testing. The Quality Assurance team can review the latest unified version and find defects. Defects can be unsatisfactory construction, non-adaptive construction, which may affect previous functionality. A recent survey shows that companies use continuous integration in their software development process because of its outstanding features and automation testing. Continuous deployment and continuous delivery are considered best practices to keep the application available at all times and to introduce new changes into production as soon as possible. This keeps the team up to speed and makes the business interactive with customers by providing them with high quality standards that can be tested automatically. Jenkins is an open-source tool written in the java programming language with built-in plugins for continuous integration. Jenkins is widely used in the

software development lifecycle, in build and test projects, helping new changes to be incorporated into the main codebase, providing customers with a new version.

2. LITERATURE REVIEW

In this [1] the continuous integration model is described and the various stages of the Continuous integration are described.[2] presents the challenges and solutions to the delivery of the software's through Build , Test and deployment stages.[3] the author performed GUI testing with various testing tools and it is concluded that use of selenium can increase test coverage.[4] talks about the DEVOPS and the pipelining the different stages and containerizing the application with Docker.
The continuous integration approaches were discussed in [5] where the best practices can be implemented in software development and achieve higher level of collaboration. Continuous delivery was also explained briefly where the users can perform various types of testing like unit test,integration testing, Acceptance testing and other types of testing to verify the product functionality and performance before releasing it to the market.
The automation can be implemented in Software development lifecycle like agile or waterfall method and can be made more reliable and faster. [6] explains all the steps in agile software development cycle.
In paper [7], the use of Jenkins in automation of continuous integration and continuous delivery was discussed and its various features and how the automation can be achieved and various types of test automation. The python-based automation was briefly explained in [8]. The reusability of software and hardware in the automation and its advantages over manual execution was pictured and effective use of resources can be made during the automation.

3. NEED FOR AUTOMATION

To reduce compatibility issues, reduce manual workload and bring software updates to users faster for a better experience. Subsequent development cycles will require repeated execution of the same set of tests. It is necessary to speed up the deployment of recent software updates for greater reliability and user responsiveness. To avoid insufficient testing or bugs in a product, a shared repository is needed where developers and operations (devops) can easily identify the type of defect. Slow-release process of software updates, identify and resolve vulnerabilities before they impact customers is time and money consuming which is not user reliable and with the demand for rapidly developed and deployed web clients there needs to be a infrastructure to account for these problems.

4. JENKINS SINGLE SERVER ARCHITECTURE

The Jenkins Master-Slave architecture is used for distributed build management. Master and slave communication via standard Internet protocol. The primary Jenkins server is the server and it performs the tasks mentioned below:

- Running builds on the slaves.
- Maintenance of slaves (software packages).
- Provides a stable version for production.
- Schedule tasks.

Jenkins slaves are remote machines tagged in the respective server. The main functions of the Jenkins slave are:

- Receive requests sent by the master.
- Slaves can run on many different operating systems.
- Slaves provide the flexibility to periodically perform specific tasks on them.
- These slaves are usually virtual machines deployed on VMware.

5. METHODOLOGY

Continuous Integration and automation Process includes following steps: Firstly, setup of source code management tool like GIT, and then installing the Jenkins, Cross-compilers, Cygwin for cross compatibility building of Cplusplus project. Installing all the required software's for building of the project and deployment.

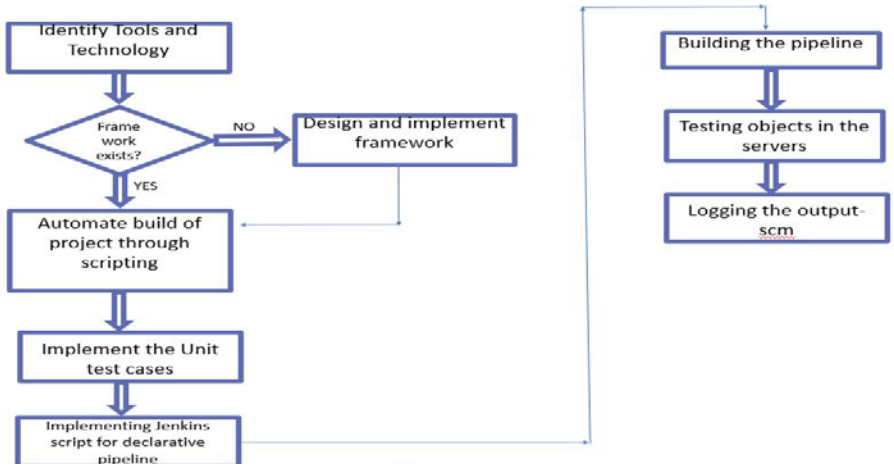

Fig -1: Flow chart

Secondly, setup the Jenkins webpage and create a pipeline job. Implement the pipeline for CI through Jenkins script for execution of different stages. Next, link the source code management tool repository to Jenkins to pull the code whenever there are code changes committed to SCM. Monitor the pipeline for different stages through Jenkins. The configuration of the same is shown in figure 2.

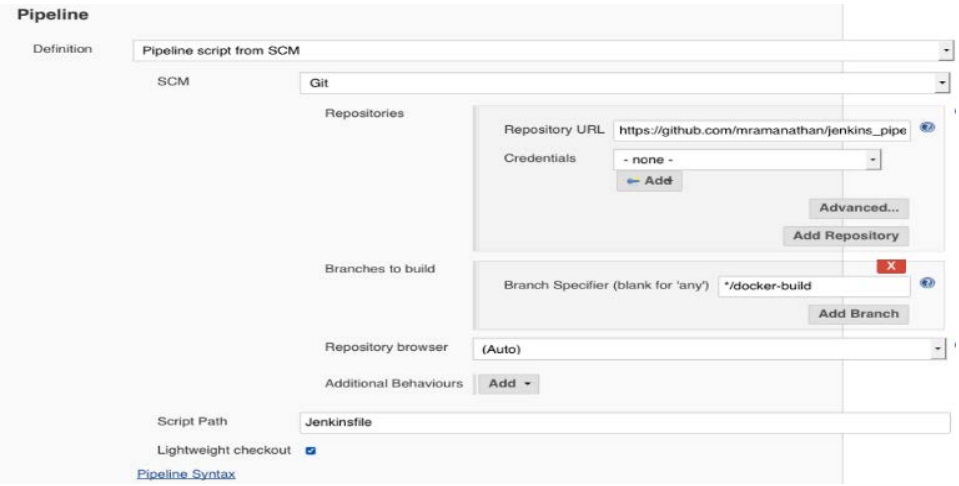

Fig -2: Pipeline job configuration

Build the project through scripting and creating the makefiles that will execute in cygwin for building the different objects created during building stage in Jenkins pipeline. Then, deploying and testing the objects in VMware by executing the different test cases or basic acceptance test cases. Finally, Logging the results to SCM. Jenkins provides all the necessary tools to implement these stages.

Maintenance of a software is an activity which includes frequent development, error corrections, optimization and deletion of existing feature, these changes may affect the system corrupted, working not properly so regression testing becomes necessary. Regression testing can be defined as the software testing technique that guarantees that recent updates made to the code is not affecting the earlier

features. There can be full or partial selection of existing and executed test cases which are runrepeatedly to assure the software stability.

6. RESULTS

The figure 3 shows the Jenkins pipeline job with different stages like declarative checkout , build, deploy and unit test. Other stages can be added as required by the users.

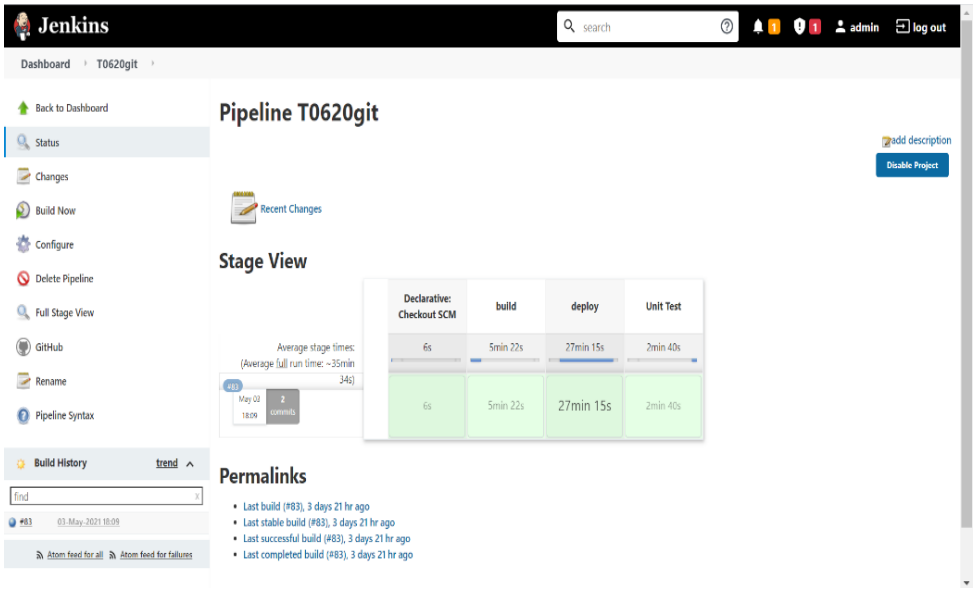

Fig -3: Jenkins pipeline

The fig 5 shows the logs of unit test stage in Jenkins. The testing was done after establishing connection to remote server through ssh. This step included many unit test scenarios which were executed by ssh commands.

```
pipeline {
    environment {
     logfile=""
    }
    agent any
     parameters {
       text(defaultValue: '0', description: 'Run specific bats', name: 'Testcase')
     }

    stages
        {
            stage('build')
        {
            steps {
                    bat(/build.bat/)
                }
                }
            stage('deploy')
        {
                steps {
                    bat(/ftptest.bat /)
                }
            }
        stage('Acceptance testing and Report Generation')
        {
                steps {
                    script
                        {
                        def
                        for (int i = 0; i <params.Testcase.size(); ++i)
                          {
                           sshScript remote: remote, script: "batsexec.sh ${params.Testcase[i]}"
                          }
                    }
                }
            }
        stage('Push to GIT')
        {
                steps {
                    withCredentials([usernamePassword(credentialsId: 'gitchekin', passwordVariable: 'PASS', usernameVariable: 'USER')]) {
                            script {
                            env.encodedPass=URLEncoder.encode(PASS, "UTF-8")
                        }
                        sh "echo ${WORKSPACE}"
                        sh '''
                        echo ${WORKSPACE}

                        '''
                        sh 'git add ${JENKINS_HOME}/workspace/T0620git/logs/${BUILD_NUMBER}/*'
                        sh 'git commit -am "checkin log for devops run [ci skip]"'
                        sh 'git push origin branch_name'

                    }
                }
            }
```

Fig -4: Jenkins pipeline script

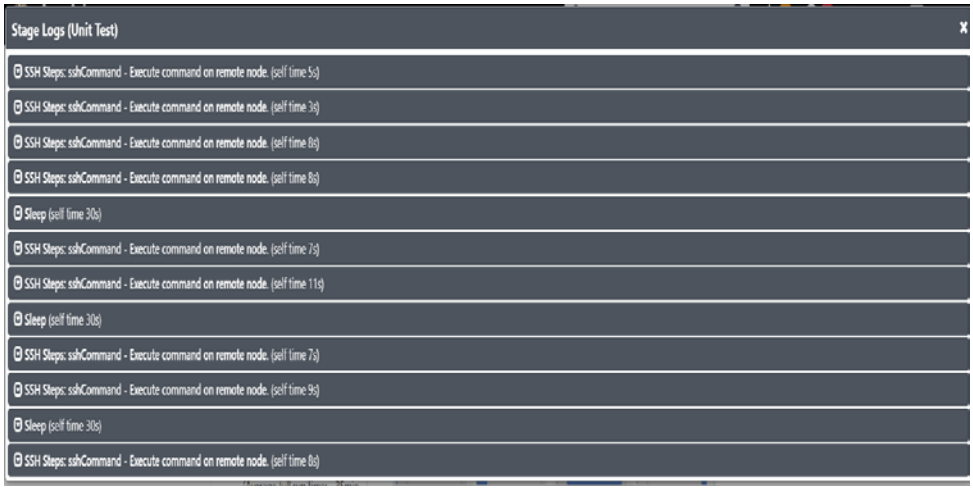

Fig -4: unit test stage logs in Jenkins

The different stages were implemented and included in Jenkins declarative pipeline script and thus the continuous integration and continuous delivery of the software was achieved as shown in fig 4.

ACKNOWLEDGEMENT

I would like to express our gratitude to our guide Sowmya Nag K for guiding us in each step of project.

CONCLUSIONS AND FUTURE SCOPE

Continuous integration and continuous delivery is an ideal scenario for application teams in an organization. Developers simply push the code to a repository. This code will be integrated, tested, deployed, retested, merged with the infrastructure, undergo quality and security audits, and be ready to deploy with extreme confidence.

Using Continuous integration and Continuous delivery, code quality is improved, and software updates are delivered quickly and with certainty that no significant changes will occur. The impact of any release can be correlated with production and operations data. It can also be used to plan for the next cycle, a key DevOps approach to the organization's cloud transition.

The software development lifecycle is revolutionized with DevOps, the cloud nativeapproach and the microservices architecture. DevOps integrates test and productionenvironments, and developers can see issues before the application goes live. ApplyingAI and ML to DevOps pipelines can help run much better builds and automation withdeeper control over information. People are moving from DevOps to DataOps and AIOps,with an emphasis on using artificial intelligence and machine learning to learn logs andmonitoring metrics to drive DevOps in a controlled manner.

REFERENCES

[1] Viktor Farcic. DevOps 2.0 Toolkit “Automating the Continuous deployment Pipeline with Containerized Microservices”

[2] Aleksi Hakli. "Implementation of Continuous Delivery Systems", Faculty Council of the Faculty of Computer and Electrical Engineering 4 May 2016

[3] J. Humble and D. Farley, Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation. Addison-Wesley, 2010.

[4] D. Stahl and J.Bosch. “Modelling continuous integration practice differences in industry software developent”, Journal of Systems and Software, vol. 87, pp. 48-59, 2014.

[5] 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.
[6] Y. Liu, Y. Lu, Y. Li, An Android-Based Approach for automatic unit test, 2015.
[7] S. Sivanandan and Yogeesha C. B, "Agile development cycle: Approach to design an effective Model Based Testing with Behaviour driven automation framework," 20th Annual International Conference on Advanced Computing and Communications (ADCOM), Bangalore, 2014, pp. 22-25.
[8] K. Jambunatha, "Design and implement Automated Procedure to upgrade remote network devices using Python," 2015 IEEE International Advance Computing Conference (IACC), Banglore, 2015, pp. 217-221.
[9] Victor E. L. Valenzuela, Vicente F. Lucena, Nasser Jazdi, Peter Göhner, "Reusable hardware and software model for remote supervision of Industrial Automation Systems using Web technologies" in IEEE, ISBN 978-1-4799-0864-6/13/\$31.00 ©2013.