iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** VII **Month of publication:** July 2022

DOI: https://doi.org/10.22214/ijraset.2022.45969

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue VII July 2022- Available at www.ijraset.com

Continuous OS Deployment for Servers

Abhishek KM[1], Prof. Somesh Nandi[2]
[1, 2, 3]*Department of Electronics and Communication Engineering, RV College of Engineering*

Abstract: ***Operating Systems have been a fundamental component of computer industry and are being constantly updated to keep on par with technological trends. These new releases, however, may have some configurations that can cause miss functioning of hardware it has been installed on. This is especially crucial in case of Mission Critical Servers (MCS) that are heavily employed for large scale applications. Manual testing of the new releases adds another dimension to existing problem especially if numerous install targets are present. In this project, an extreme automation testing solution is developed to completely circumvent woes incurred due to manual testing. The proposed solution also takes care of various corner cases, which otherwise would be difficult to detect.***
Keywords: Qcow2, automated pipelines, Continuous OS deployments, Mission Critical Servers, Jenkins

I. INTRODUCTION

Updation of Operating Systems (OS) has been a historical trend which has become increasingly aggressive in current years. Companies have reduced the product lifecycle to better fit the rapidly changing technological trends and subsequently gain edge over its competitors. However, compatibility of new releases and induced behavioral changes of system under test due to new releases needs to be thoroughly investigated. The influx of new OS releases and vast number of systems to be tested, render manual testing highly excruciating and inefficient. OS Deployment is the process of deploying OS images that are captured to target computers. From fixing certain system-related issues—like hard disk failure, data recovery, and system slowdown—to providing a device to a new employee, OS imaging and deployment is one of an IT admin's primary tasks.

Along with this saving resources like time is a major challenge faced by the developing companies, so automation in integration and deployment using Jenkins ansible saves a lot of time and also if there are changes which need to be done in the project frequently, it can be easily updated [1]. Most of the projects that use automation are agile projects and to manage them CCID is used [2]. In agile, new features are introduced to the system in each sprint delivery, and although it may be well developed, the delivery failures are possible due to performance issues. By considering delivery timeline, moving for system scaling is common solution in such situations. Jenkins is implemented in a master/slave architecture where master node is the Jenkins server and slaves are the Jenkins clients [3]. However, Jenkins has evolved from being a pure Continuous Integration Platform to a Continuous Delivery one, embracing the new design tendency where not only the build but also the release and the delivery process of the product is automated [4]. Opensource and commercial testing tools that may help users to select the appropriate software testing tool based on their requirements are present on the internet [5] [6]. These tools have comprehensive comparisons with other tools as well [7]. However, only implementation is not sufficient and necessary reports for debugging are required as well. [8] present a method of automatic analysis of the test reports generated after software integration testing by utilizing text mining techniques. Text mining is a data analysis technique employed to elicit high-quality information from textual sources like full-text documents, emails, and HTML files. The total product quality and efficiency can be greatly increased in the major software development processes managed with versioning systems, with versions delivered much faster, less staff assigned for manual testing and fewer software errors emerge [9].

In the era of software productions, every software should have a certain quality that it assures and this quantization is given in [10]. Work has been done on creating a proof of concept for designing an effective framework for continuous integration, continuous testing, and continuous delivery to automate the source code compilation, code analysis, test execution, packaging, infrastructure provisioning, deployment, and notifications using build pipeline concept [11]. There has been an effort to implement CI/CD in the performance test. In the existing test, the test still needs humans to conduct the test. Our proposed solution for performance tests with CI/CD can be performed automatically and reduce human role in the test [12]. This partially solved using cloud technology and [13] has a brief cloud technology overview and guideline on how cloud computing technology is transforming industrial automation for the future, enhancing productivity and cost optimization. Continuous integration of specific information systems, a collaborative work scheme for continuous integrated delivery based on Jenkins and Ansible is required [14].

©IJRASET: All Rights are Reserved

An all in all solution to create an entire automated pipeline, starting with detecting changes in the Java-based web application source code, creating new resources in the Kubernetes cluster to host this new version and finally deploying the containerized application in AWS is done [15].

Typically, Disk imaging software is used to create an image of the OS and disk partitions of the disk in computer. After imaging a hard disk, image can be stored in the image repository configured in user environment. The image created using OS imaging can then be deployed to a bare-metal computer, computer with a corrupted OS, or even to multiple computers simultaneously [16]. Hard drive imaging software supports both online and offline imaging modes to image a computer. Thus, using comprehensive disk imaging software we can onboard huge number of workstations with minimal manual effort. This method allows administrators to capture the disk image of a system when it is live and functioning in the network. This hard drive imaging method does not require a system reboot to perform imaging [17]. It finishes the work, without interrupting the user and their ongoing work. Administrators can use this hard disk imaging method to acquire the disk image of a system in the network that is shut down. When this method is used, the system is booted into the Windows pre-installation environment, and disk imaging is performed using the WinPE component. This method is called offline imaging [18]. When deploying operating systems for five computers or less, administrators can opt for a unicast method, whereby the image is taken and sent to each computer one by one. To deploy operating systems on multiple computers, administrators can opt for the multicast method, where the image is taken and sent to all the computers simultaneously [19]. While choosing multicast method, administrators can also save the bandwidth usage by configuring the appropriate settings.

Hence, it can be inferred that automation plays a very crucial role in easing the usage of any developed system. A system with highly automated functionalities exhibits versatile usage. It is also inferred that Jenkins is the best tool available in current market to automate processes.

II. DESIGN CONSTRAINTS AND METHODOLOGY

Continuous Qual OS Deployment was developed to be end to end automated with no manual intervention for the entire cycle of execution. The design aim of the project was also to identify any faults in 3 areas. First, server machines are constantly subjected to changes either by users or firmware/software updates that are initiated by administrators. Such changes can trigger a cycle of fault points that not only need to be identified but also rectified at earliest. As a matter of fact, these changes can be observed only if server machines encounter certain settings or configurations. Second, companies have specialized server machines that are used to run OpenStack ironic services to deploy OSes. These server machines or alternatively called, sites are constantly updated and restarted to support latest releases. In process, some previously running services maybe paused or stopped entirely. For this paper, Qcow2 images are used to deploy on machines. These Qcow2 images are created from base images while customizing to fit certain requirements. These customized settings, unknown to user, may be an impediment to successful deployments.

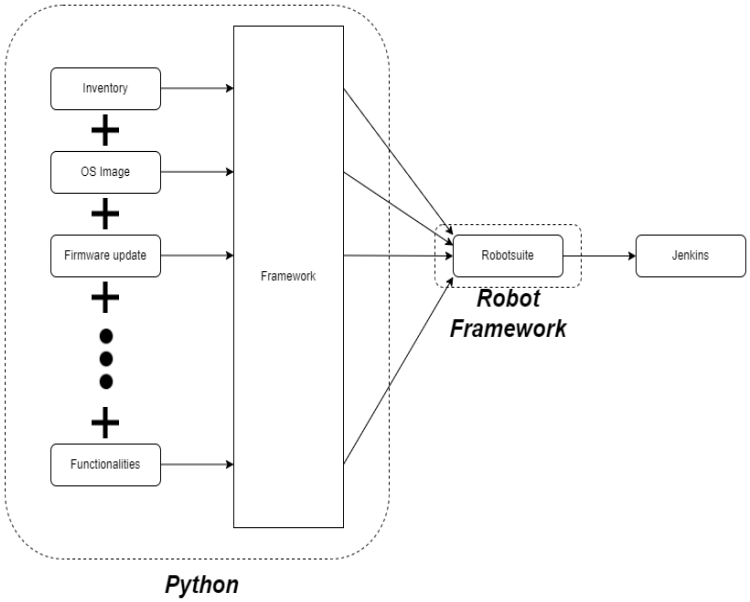


Figure 1: methodology flow

Figure 1 depicts the methodological order of event that is taken to create the hardware configuration tests. Single functionality is first developed in python. These include usage of commands that are utilized for controlling the server health parameters. These functionalities may include, power cycling the partition, or resetting partition and so on. A number of such scripts are written for different and disjoint functionalities. In order to test a server's capabilities to handle multiple functionalities concurrently, the individual scripts are wrapped in robosuite. Jenkins is used to run and test the individual as well as robosuite scripts. Jenkins, implements the commands on the server, real-time and logs in the results. Thus, providing very accurate information about the health of servers.

III. IMPLEMENTATION

Implementation of this paper is done in two phases that have been executed parallelly and are given below

A. Qcow2 image creation

The first step to realize this project was Qcow2 image creation as show in Figure 2.

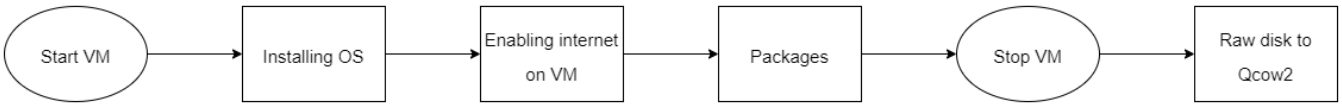


Figure 2: Qcow2 image creation flow

This process is same for SUSE Linux, Redhat Linux, Oracle Linux, Windows and VMWare ESXi operating systems. The Virtual Machine should prior be installed with virt-manager. A new virtual manager should be created using ISO file for the corresponding Qcow2 image. After customizing the settings, a raw image of format .IMG is created and booted to UEFI. The internet will then be enabled to download required packages on the raw image. After the repos have been enabled, the virtual manager is shutdown and the raw image is converted to Qcow2 format.

B. Continuous OS deployment

Figures 3 and 4 depict the flow of process for Continuous OS Deployment. To ensure end to end automation, the algorithm itself selects and reserves the server for 180 minutes. However, some server machines might be in critical health or undesirable run state which renders it unsuitable for testing.

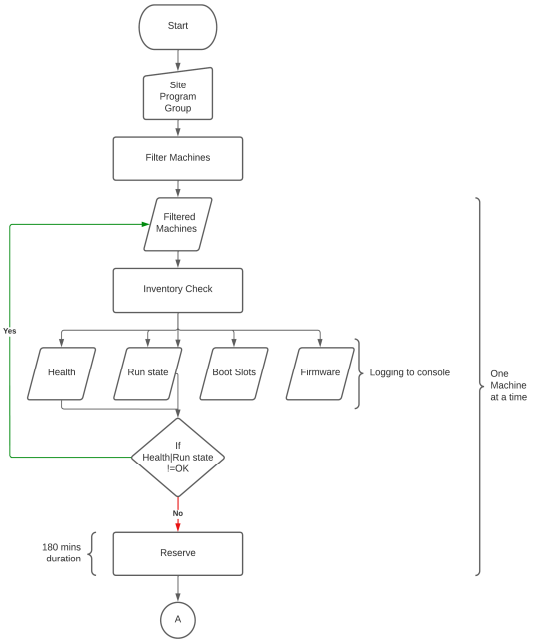


Figure 3: Algorithm flow for one-time jobs

To skip such machines, an inventory check is performed that will fetch Health, Runstate, Boot slots and Firmware versions. If machine has desirable state then it reserved and used further. Firmware is updated the server to latest configurations and skip the update in case the machine is already updated. A predefined list of Qcow2 images is declared as per user requirements and each Qcow2 image is deployed on machine. OS verification verifies successful deployment and logs out the IP address that is given to OS image creation. OS image contains the OS version and bootloader path in UEFI Shell. Finally, necessary information is uploaded to a dashboard so that it can be viewed by engineers and clients alike.

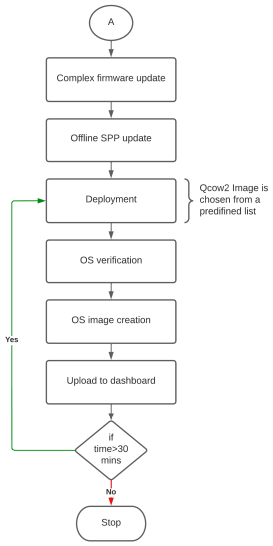


Figure 4: Algorithm flow for repeated jobs

As it is evident that the OS and machine will be continuously tested during entire cycle, the project measures quality of targets and hence, is called Continuous OS Deployment.

IV. RESULTS

Results and related discussions form the basis for the outcomes of any project. Jenkins platform is used to run the developed project and it itself provides comprehensive report for debugging purposes. Since Continuous OS Deployment is an amalgamation of different test cases, at the end of each run, Jenkins provides a summary report comprising the status of each test case as shown in Figure 5

```
17:47:52  ------------------------------------------------------------------------------
17:47:52  Inventory_check :: Inventory check of Machine                      | SUCCESS |
17:47:52  Test execution successful !!
17:47:52  ------------------------------------------------------------------------------
17:47:52  Firmware update :: Update the firmware                            | SUCCESS |
17:47:52  Test execution successful !!
17:47:52  ------------------------------------------------------------------------------
17:47:52  Offline_SPP :: Offline SPP on the reserved proto                   | SUCCESS |
17:47:52  Test execution successful !!
17:47:52  ------------------------------------------------------------------------------
17:47:52  Deploy_And_Verify_OS :: Deploying a qcow2 image and verifying the OS  | SUCCESS |
17:47:52 Test execution successful !!
17:47:52  ------------------------------------------------------------------------------
17:47:52  OS deploy qual :: Run multiple tests on host                       | SUCCESS |
17:47:52  5 tests, 5 passed, 0 failed
17:47:52  ==============================================================================
17:47:52  Debug:    /home/jenkins/logs/misc/2022/07/17/run_robot_suite_10791/robot_debug-20220717-061750.log
17:47:52  Output:   /home/jenkins/logs/misc/2022/07/17/run_robot_suite_10791/output-20220717-061750.xml
17:47:52  Log:      /home/jenkins/logs/misc/2022/07/17/run_robot_suite_10791/log-20220717-061750.html
17:47:52  Report:   /home/jenkins/logs/misc/2022/07/17/run_robot_suite_10791/report-20220717-061750.html
```

Figure 5: Summary report

In case of failure of any test case, engineers can refer to debug, output, log or report files present in Figure 5 as well.
The project is a result of many iterations over entire period of development. This progress can be tracked by Jenkins as shown in Figure 6 that plots the successful or failed run with each build.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

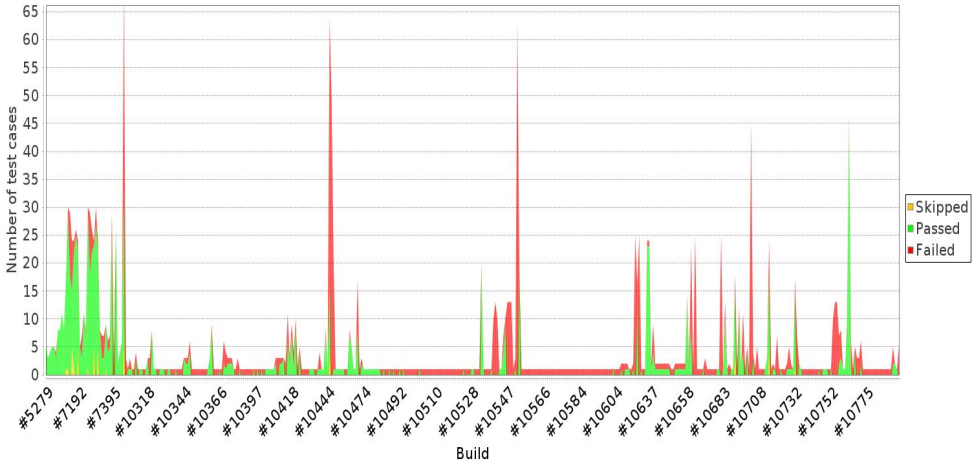


Figure 6: Build status for each run

Here it can be inferred that builds 10752 has maximum number of passed test cases, hence best suited for running Continuous OS Development. After successful test run, the Qcow2 image details and system configurations is updated on a dashboard so that it can be viewed by engineers and clients alike. A prototype of this dashboard is shown in Figure 7

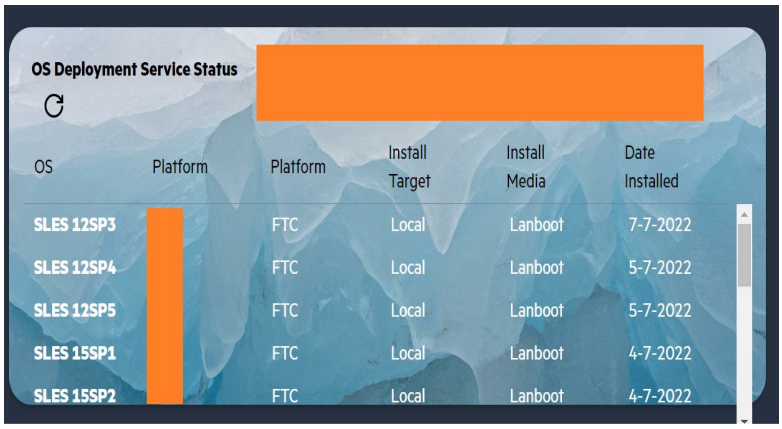


Figure 7: Web solution prototype

The results obtained after many iterations and successful run were shown and discussed in this chapter. Having obtained good results, it is very important to discuss on the conclusions that can be made through the entire project, future scope and learning outcomes of the project.

V. CONCLUSIONS

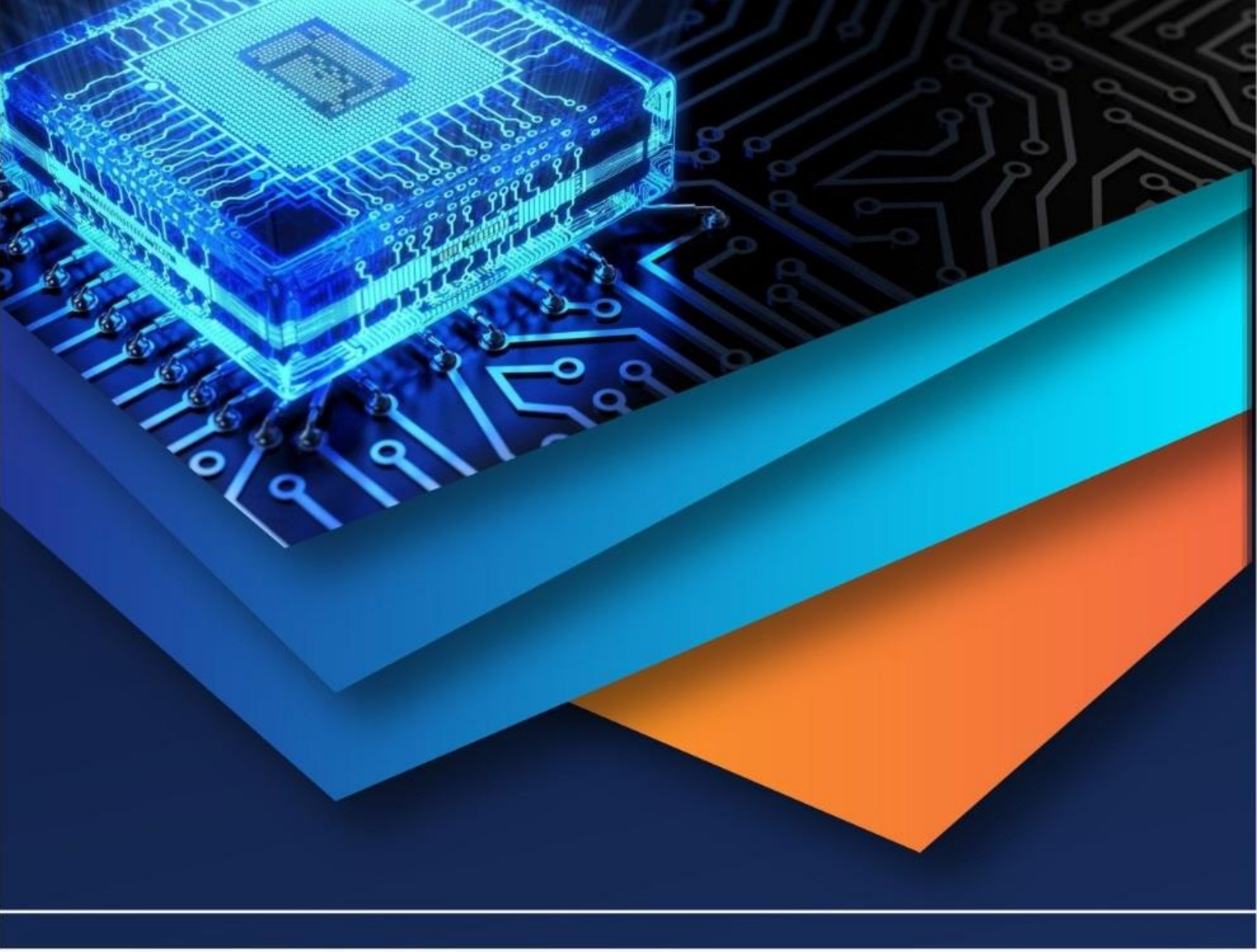
Continuous OS Deployment is an extreme automation project that provides seamless quality check for OS and server machines. The Qcow2 images have been developed using Openstack Ironic services. Abundance of documentation and easy integration with environment makes Openstack ironic services extremely suitable for this purpose. Continuous OS Deployment is created using Robot Framework. It has been developed by realizing configurations to be implemented using python and then wrapped together using Robotsuite. Manual implementation of the steps described in previously is extremely excruciating and time consuming. Continuous OS Deployment serves as a replacement to this and takes care of corner cases that, if done manually, would have been very hard to debug

VI. ACKNOWLEDGMENT

We thank all members from OpenStack development team, members from hardware system testing and OS enablement team and business IT admins across sites in helping us integrating various tools and deployment of the eco-system and their continued support from time to time. We also thank RV College of Engineering, ECE Department for their unwavering support and coordination that enabled us to realize this project

REFERENCES

[1] S. A. I. B. S. Arachchi and I. Perera, "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management", Moratuwa Engineering Research Conference (MERCon), 2018
[2] N. Seth and R. Khare, "ACI (automated Continuous Integration) using Jenkins: Key for successful embedded Software development", 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS), 2015
[3] S. Mysari and V. Bejgam, "Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible", International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), 2020
[4] V. Armenise, "Continuous Delivery with Jenkins: Jenkins Solutions to Implement Continuous Delivery", IEEE/ACM 3rd International Workshop on Release Engineering, 2015
[5] P. A. G. Permana, E. Triandini and N. L. G. P. Suwirmayanti, "Implementation Jenkins Automation Deployment with Scheduler and Notification", 3rd International Conference on Cybernetics and Intelligent System (ICORIS), 2021
[6] Heidilyn V. Gamido, Marlon V. Gamido, "Comparative review of the features of automated software testing tools", International Journal of Electrical and Computer Engineering (IJECE),Vol. 9, No.5, 2019.
[7] Vinita Malik, Mamta Gahlan, "Comparative Study of Automated Web Testing Tools", International Journal of Latest Trends in Engineering and Technology (IJLTET), 2016
[8] F. Okezie, Odun-Ayo and S. Bogle, "A Critical Analysis of Software Testing Tools", International Conference on Engineering for Sustainable World, 2019
[9] P. Paigude, V. Gajul, J. Mishra and S. Katkar, "Software Integration Test Report Analysis Automation Using Python", Asian Conference on Innovation in Technology (ASIANCON), 2021
[10] A. Mori, "Anomaly Analyses to Guide Software Testing Activity", IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), 2020
[11] E. Çelik, S. Eren, E. Çini and Ö. Keleş, "Software test automation and a sample practice for an enterprise business software," International Conference on Computer Science and Engineering (UBMK), 2017
[12] Y. Zhao, Y. Hu and J. Gong, "Research on International Standardization of Software Quality and Software Testing", IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall), 2021
[13] M. Soni, "End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery,", IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), 2015
[14] M. R. Pratama and D. Sulistiyo Kusumo, "Implementation of Continuous Integration and Continuous Delivery (CI/CD) on Automatic Performance Testing", 9th International Conference on Information and Communication Technology (ICoICT), 2021
[15] S. Garg and S. Garg, "Automated Cloud Infrastructure, Continuous Integration and Continuous Delivery using Docker with Robust Container Security", IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 2019
[16] W. Yiran, Z. Tongyang and G. Yidong, "Design and implementation of continuous integration scheme based on Jenkins and Ansible", International Conference on Artificial Intelligence and Big Data (ICAIBD), 2018
[17] M. H. Alkawaz and A. Silvarajoo, "A Survey on Test Case Prioritization and Optimization Techniques in Software Regression Testing", IEEE 7th Conference on Systems, Process and Control (ICSPC), 2019
[18] client-side web applications", International Electrical Engineering Congress (iEECON), 2017
[19] A. Cepuc, R. Botez, O. Craciun, I. -A. Ivanciu and V. 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 (RoEduNet), 2020



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)