Applying Unsupervised Machine Learning in Continuous Integration, Security and Deployment Pipeline Automation for Application Software System

Deepak Raj D S, Swarnalatha P

Abstract: ***Continuous integration and Continuous Deployment (CICD) is a trending practice in agile software development. Using Continuous Integration helps the developers to find the bugs before it goes to the production by running unit tests, smoke tests etc. Deploying the components of the application in Production using Continuous Deployment, using this way, the new release of the application reaches the client faster. Continuous Security makes sure that the application is less prone to vulnerabilities by doing static scans on code and dynamic scans on the deployed releases. The goal of this study is to identify the benefits of adapting the Continuous Integration - Continuous Deployment in Application Software. The Pipeline involves Implementation of CI – CS - CD on a web application ClubSoc which is a Club Management Application and using unsupervised machine learning algorithms to detect anomalies in the CI-CS-CD process. The Continuous Integration is implemented using Jenkins CI Tool, Continuous Security is implemented using Veracode Tool and Continuous Deployment is done using Docker and Jenkins. The results have shown by adapting this methodology, the author is able to improve the quality of the code, finding vulnerabilities using static scans, portability and saving time by automation in deployment of applications using Docker and Jenkins. Applying machine learning helps in predicting the defects, failures and trends in the Continuous Integration Pipeline, whereas it can help in predicting the business impact in Continuous Delivery. Unsupervised learning algorithms such as K-means Clustering, Symbolic Aggregate Approximation (SAX) and Markov are used for Quality and Performance Regression analysis in the CICD Model. Using the CICD model, the developers can fix the bugs pre-release and this will impact the company as a whole by raising the profit and attracting more customers. The Updated Application reaches the client faster using Continuous Deployment. By Analyzing the failure trends using Unsupervised machine learning, the developers might be able to predict where the next error is likely to happen and prevent it in the pre-build stage.***

Keywords: Continuous, Integration, Security, Delivery, Docker, Registry, Jenkins, Pipeline.

I. INTRODUCTION

The Software Development Life Cycle (SDLC) is the process followed for an application software which includes plans describing how to design, develop, build, test and deploy an application. Software Companies want their new releases / updates to be delivered to the clients in fast and secure way as possible. In the traditional way, after the development process, quality of the code is tested manually by running various tests in the different environments, building the components of the application and deploying them to production which consumes a lot of human time and effort in executing these steps. The Continuous Integration process improves the code quality, makes the software bug free and reliable, while the Continuous Deployment process improves the software release time to the customer, improves the deployment process time of components in applications, availability of the components throughout the production datacenters which can be located on the other side of the world. There are wide range of tools offered for the Continuous Integration Pipeline (Eg. Jenkins, Travis CI) which could automate the testing processes. There might be components that have to be built in order for the quality assurance teams to validate them, these CI tools can help pre-build those so that they can be tested before the deployments. Continuous Security ensures that the code doesn't any security flaws which might expose the Data of the clients, Algorithm behind code, out dated libraries if any etc. After the scans are done for Continuous security, the tools which are used to tests these, offer remedies to the issues and these can be fixed in pre-release. Machine learning has been used in various fields in current world, may it be health care, financial services, marketing etc. Similarly, it can also be used in the software development life cycle to improve the efficiency of the DevOps cycle. It can be used to find trends which will help predict its immediate impact before the next phase and enables us to proceed to the next step or rollback to the previous versions.

II. PROPOSED METHODOLOGY

In this research, the author uses Pipeline by implementing Continuous Integration using Jenkins Tool for run tests, the author will use various internal modules to run tests depending upon the language used to develop the code, since this application is built using node.js, the author will use jshint package to test the quality of the code. A testing framework called Jasmine will be used to run the unit tests. Jenkins is an open source automation server written in java used for CICD processes. Jshint is a static code analysis tool used to check the code

Revised Manuscript Received on November 15, 2019

Deepak Raj D S, Software Engineer, Netskope, Bengaluru, India.
Swarnalatha P, Associate Professor, Department of Information Security, SCOPE, Vellore Institute of Technology.

quality detecting syntax errors and potential problems in node.js scripts. Jasmine is static code testing tool used to run unit tests for the same. The continuous security is done by running static scans using a tool called Veracode which generates report containing security flaws of the code and remedies to fix them. Veracode is a cloud-based security scan tool used to run static and dynamic scans, mobile application behavioral analysis and software composition analysis. For continuous deployment the author will use Docker along with Jenkins for deployment. Docker is a set of platform-as-a-service products that provides operating system level virtualization to deliver software packages called containers. By using these tools in different stages development of the Application Software, the benefits of adapting the CI – CS – CD process is analyzed using the Pipeline. The Pipeline consists of a Web Application is built using a server-side script language Node.js, this web application also uses Amazon Web Services (AWS) S3 as a to store Images information, Mongo is used as the Backend Database for the application. As the testing phase happens in the continuous integration process, the data related to the success or failure of the tests will be stored which can be used to predict / analyze using machine learning when something changes from the regular trend both for the good and the bad.

A. Research Questions

1. Once the development is complete, will implementing CICD shorten the release time?
2. How does implementing CICD process affect the Error Rate, Bugs identified, Infrastructure Costs?
3. Can Machine learning be used to compare performance characteristics for tests run in Continuous Integration?
4. Can Impact of the new version of artifacts be predicted based on the trend?

III. THEORETICAL BACKGROUND

A. Continuous Integration

Continuous Integration (CI) is a development practice where developers push their code into repository. Each integration

B. Continuous Security

the Messenger platform Tool, the controllers for them are termed as components in the automation.

C. Continuous Deployment

Continuous Deployment is a step up from Continuous Delivery in which every change in the source code is deployed to production automatically, without explicit approval from a developer. A developer's job typically ends at reviewing a pull request from a teammate and merging it to the master branch. A CI/CD service takes over from there by running all tests and deploying the code to production, while keeping the team informed about outcome of every important event. [3]Continuous Security is the addressing of security concerns and testing in the Continuous Deployment pipeline, and is as much a part of Continuous Deployment as operations, testing, or security is a part of the DevOps culture. It can also be used as part of Continuous Integration Pipeline.[2] can then be verified by an automated build and automated tests. The tests are automated using various CI tools.[1]

D. Unsupervised Machine Learning

This method of ML finds its application in areas were data has no historical labels. Here, the system will not be provided with the "right answer" and the algorithm should identify what is being shown. The main aim here is to analyze the data and identify a pattern and structure within the available data set. Transactional data serves as a good source of data set for unsupervised learning.

IV. AUTOMATION PIPELINE OF CI-CS-CD

A. Introduction

The Author uses an application that he developed named `ClubSoc` which is used for managing club activities in his university.

B. Overview of the CI – CS – CD -process

The Whole CI Process is implemented in Jenkins where once code is committed to the GitHub, the push event triggers the CI Pipeline in Jenkins, which then checks out the version and runs the unit tests, component tests, once the tests are successful , only then , the pipeline proceeds to the next phase i.e CS Pipeline. In this pipeline, the source code is set up for scanning for vulnerabilities and the report if generated, if there are any critical vulnerabilities, the pipeline exits with a state 'Failed', else the pipeline proceeds to the CD Pipeline. The source code and other executables required for the application (components) are built as docker images and pushed to the docker registry, which is then deployed in the staging machine for component testing and dynamic scan analysis. The Project follows the Model – Views – Controllers (MVC) Framework, where the Model defines the DB Schemas that used, the Views represent the UX part of the application and the Controllers represent the backend code of the application which is built using Node.js Once the planning phase is sorted out for the development, we divided the features into components, User Dashboard UX, the Admin Dashboard UX, the Events Feed UX and the Bot UX are clubbed as `Web UI` Component. The Controllers for each of those are clubbed as different components

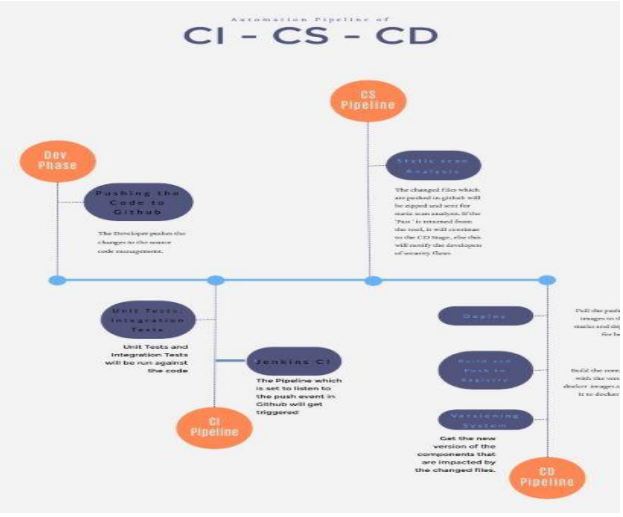

Figure 1 – CI – CS – CD Flow Chart

C. Implementing Continuous Integration

After a new feature or a component is developed, the code is pushed to GitHub which is the Source Code Management for the application, the Jenkins CI job which is listening to the push event in GitHub gets triggered and CI Integration Pipeline get triggered automatically. The Unit Tests are run for the affected components are run in parallel. Similarly, for Components tests. Since this application is built using node.js, a Unit Test Framework called Jasmine is used to run the tests, if any tests cases fail, the job will fail notifying the developers and the pipeline will be blocked. Once the tests are passed, it will the Continuous Security Pipeline as a Downstream Pipeline.

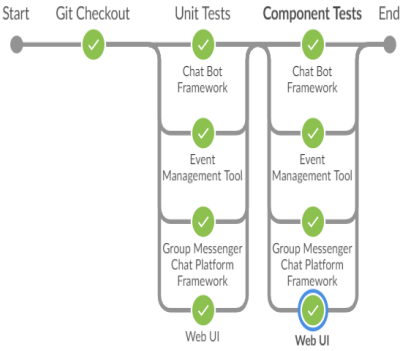

Figure 2 Continuous Integration Flow Chart

D. Detecting Quality Regressions using Unsupervised Learning

While testing the CI Tests in Jenkins, loads of logs will be stored, the data is used from the logs to analyze the trends in tests. The Unsupervised learning algorithm k-means is used to understand the frequency of events in the CI process. This is being done by forming "clusters" of related event data. Splunk have been used here to continuously verify the application quality by analyzing the log event data.

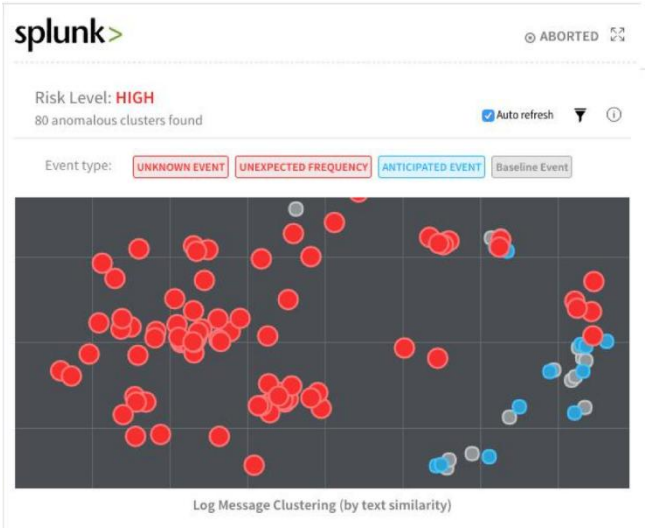

Figure 3 Splunk Tool to Identify Anomalies

In the above output shown, the grey dots represent the normal / regular trend of events in the CI Testing process while the red dot show the unusual events or new events that are encountered for the first time or the events are happening with an unusual frequency.

E. Implementing Continuous Security

The Continuous Security Pipeline consists of Static Scans which will report the flaws in the source code of the application. In this web application, the author has used Veracode application for running static scans. Once the CI Pipeline is complete, the Continuous Security Pipeline will be triggered. The Changed files of that features which are part of the push in GitHub are bundled together and sent to Veracode Platform for Scanning, the detailed report consists of flaws and recommendations for the developers to fix the flaws. Based on the flaws, a score will be provided which we set the limit of pass or failure, if the score doesn't pass the threshold score, then the CS Pipeline will fail, else this pipeline will generate the reports and trigger the CD pipeline. Below is the Sample scan result when one of our features were developed and the code was scanned for security issues, we found that there were node packages that were outdated and there has been an update on those packages, we fixed it and updated the code.

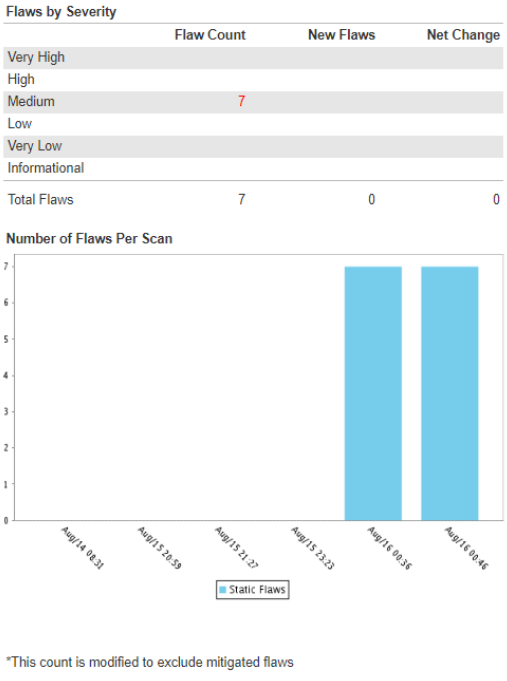

Figure 4 Veracode Report

F. Versioning System of Components

Versioning System is a component as part of the deployment process. The Author developed this in order to maintain individual versions for components so that when there has been a crash in the latest version, they can be reverted back to the previous version immediately. The Pipeline requests the Component Version using REST Call to the API Gateway, which triggers the Lambda function which returns the updated version of the component.

G. Implementing Continuous Deployment

The CD Pipeline is triggered after the CS Pipeline, using

the version system, the pipeline gets the version of the components which are then built. For each component, the author has written a dockerfile which defines the commands to be executed, packages needed for the component to run and the source code of the component. After the build process, the images are tagged using the version and are pushed to Docker registry which is a repository for storing docker images, this is called the push process. We had stacks which means a collection of servers, databases that are used for the application, the pushed docker images are then pulled and deployed in the stacks.

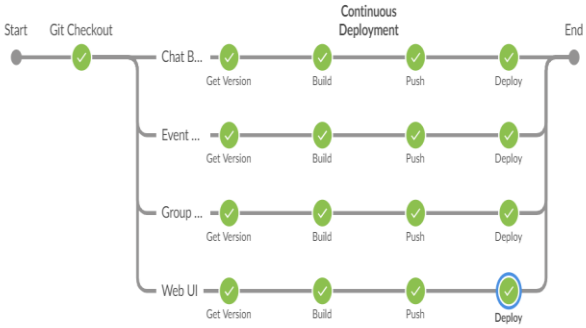

Figure 5 Continuous Deployment Flow Chart

H. Detecting Performance Regressions using Unsupervised Machine Learning

Analyzing the performance trend of each services / components after being deployed helps us identify the impact it will have on the customers, the metrics analyzed here is the response time of each component after it is deployed. SAX and Markov algorithms are used to analyze the performance time of the deployed components. Below is the difference between the performance time of the previous version of the component deployed and performance time of the current version of the same component.

	Stalls	Response Time	Error
Chat Bot Framework	✓	✗	✓
Event Management Tool	✓	✓	✓
Group Messenger Chat Platform	✓	✗	✓
Web UI	✓	✓	✓

Figure 6 Output of Applying SAX and Markov

In the above figure, the response time is marked as passed if it is same compared to the previous version of the deployed component while, it is marked as failure if the response time is marked as failure in the current deployed version of the same component.

I. Results & Findings

By Implementing the Continuous Integration – Continuous Security – Continuous Deployment process, it was found to be very useful get the latest changes deployed instantly to the clients / users of the application where they are assured of less bugs / issues per release, less prone to security vulnerabilities, and get the application update as soon as the fix or a feature is ready. By using Unsupervised Machine Learning Algorithms in Quality Regression Analysis and Performance Regression analysis, we are able to identify is there is any deviation from the normal trend. We found two deviations from the normal trend (compared to previous versions) in Performance regression analysis, which might impact the client if found in bigger applications and hence this analysis will help us in fixing this before the client reports this anomaly.

V. ANALYSIS OF IMPLEMENTING CI – CS – CD

The goal of this study was to implement CI – CS – CD process in software development, in order to accomplish this, the author implemented the process in a web application to show the process and the results of each of the processes. When a new feature is developed, it is automatically deployed and are quickly is available to the clients with more quality assurance, reduced security flaws, less bugs. When this process is implemented to an application developed by small to medium sized companies, they are assured of shorted release cycles, critical issues are detected early and fixed in the same release cycle, less security flaws which gives the clients assurance against hackers who are trying to

misuse them. In this Pipeline , there was request from of the users, that they would need more facilities with the event management application, which was to notify the user with remainder about the events that were to occur as a WhatsApp or text message, the author implemented this feature and by using this Pipeline , the updates were successfully deployed and we found that more user count has significantly increased. Few Outdated packages that were being by the source code were found during the continuous security, using the static scan reports the criticality of each issue were found and was fixed in the next release.

VI. CONCLUSION

Implementing CICSCD improves the application development process significantly. The release cycle was found to be shorter and thus it helps improving the productivity of the system. The Goals of the CI, which is to provide higher quality code by running Unit and Component tests against the code and fixing bugs, issues pre-release itself which enables ability to compete in the marketplace. By applying the unsupervised machine learning algorithms in the Continuous Integration process, we were able to identify the failure pattern and finding the quality regressions by using the logs. Using the Continuous Security Pipeline, the developers are able to find the security issues, which can prove to be very critical. Without security scans, if the code is vulnerable to the potential hacker, the client information could very well be hacked and the company had to pay the huge price of data breach and loss of clients. Smaller releases are low risk and lessen the cognitive load. Since there are shorter release cycles, even if there is a bug found post deployment, the fixes can be provided to the customer in less time. The Code in production or the new features which is in production are more money making than

to be queued waiting to be deployed. The algorithms were also helpful in the Performance Analysis after the Continuous Deployment process. This helps us in identifying the response time in each component post deployment and if there is any business impact by those. Hence, Machine Learning can quantify the business impact in the production deployment of application enabling us to find anomaly before the customer / client does. The analysis of implementing CICD in the Pipeline answers the research questions stated above, implementing CICD significantly reduces the release cycle. Using the concept of containerization, the infrastructure costs are reduced, which if not used could cost the company lost of investment in buying servers, hardware etc. The delay from when the code is written for a feature to running in production data centers is the time to value and is a bottleneck for many organizations. Continuous Delivery can help overcome this barrier to quick deployments. Error rates and infrastructure costs can be quickly and easily measured once the CICD is implemented. Implementing the unsupervised machine learning algorithms in Continuous Integration process gives us the irregular trends found while quality testing which is useful when exceptions and errors in an application is critical, using these, the errors can be easily fixed.

REFERENCES

1. Suyash Dubey, Continuous Integration with Jenkins. Available at: https://www.pcloudy.com/continuous-integration-with-jenkins **(2019).**
2. Mphasis Stelligent, Continuous Security in Continuous Delivery Pipeline. Available at:
3. https://stelligent.com/2016/04/05/continuous-security/ **(2016).**
4. Shahin, Mojtaba; Ali Babara, Muhammad; Zhu, Liming. "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices". IEEE Access. **(2017).**
5. Shahin, Mojtaba; Ali Babar, Muhammad; Zahedi, Mansooreh; Zhu, Liming. Beyond Continuous Delivery: An Empirical Investigation of Continuous Deployment Challenges. Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement **(2017).**
6. Fowler, Martin, Continuous Integration. martinfowler.com (2006).
7. Samy Bengio, An Introduction to Statistical Machine Learning - Hidden Markov Models - https://bengio.abracadoudou.com/lectures/old/tex_hmm.pdf **(2016).**
8. Pavel Senin, Symbolic Aggregate Approximation (SAX). https://jmotif.github.io/sax-vsm_site/morea/algorithm/SAX.html **(2016).**

AUTHORS PROFILE

Deepak Raj D S., B.Tech(Computer Science) is currently working as Software Engineer at Netskope, India having recently graduated from Vellore Institute of Technology (VIT). He attended his first conference which is 2nd World Summit on Advances in Science, Engineering, and Technology (IndianaSummit 2019) Organised by VIT, India at Indiana University - Purdue University, Indianapolis, USA where he presented a paper.

Swarnalatha P., Associate Professor, Department of Information Security, SCOPE, Vellore Institute of Technology.