International Journal of Computer Science and Software Engineering (IJCSSE), Volume 8, Issue 10, October 2019
ISSN (Online): 2409-4285 www.IJCSSE.org Page: 247-253

Continuous Integration – Continuous Security – Continuous Deployment Pipeline Automation for Application Software (CI – CS – CD)

Deepak Raj D S[1] and Swarnalatha P[2]

[1] B.Tech (Computer Science), VIT University, Vellore
[2] School of Computer Science and Engineering (SCOPE), VIT University, Vellore

ABSTRACT

Application software which are developed by software companies around the world have clients who demand more from the companies which can be new features developed, bugs fixed, improvements made etc., Continuous integration and Continuous Deployment (CICD) help deliver these to the clients in more reliable and faster way. It also helps the developers in testing their new features as soon as they build them using Continuous Integration. Deploying the components of the application in Production using Continuous Deployment, this way the new release of the application reaches the client faster than the ones who don't implement CICD. 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 Security – Continuous Deployment in Application Software. The CI – CS – CD process is implemented on a web application ClubSoc which is a Club Management Application which the author developed. The results have shown by adapting this methodology the author is able 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.

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

1. INTRODUCTION

The Software Development Life Cycle 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.

2. RESEARCH METHODOLOGIES

In this research, the author uses his project 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 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 project. The project 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.

2.1 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. THEORETICAL BACKGROUND

3.1 Methodologies

3.1.1 Continuous Integration

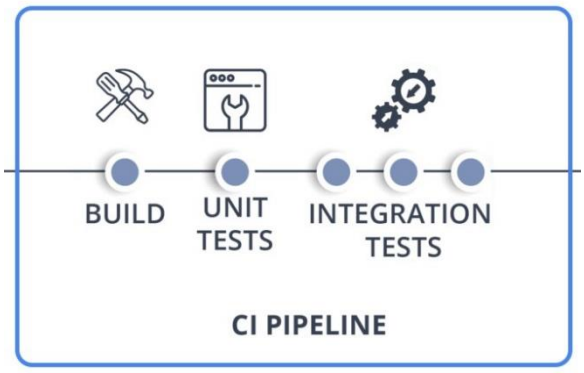

Fig. 1. Continuous Integration (CI)

Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day. Each integration can then be verified by an automated build and automated tests. While automated testing is not strictly part of CI it is typically implied.

3.1.2 Continuous Security

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

3.1.3 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.

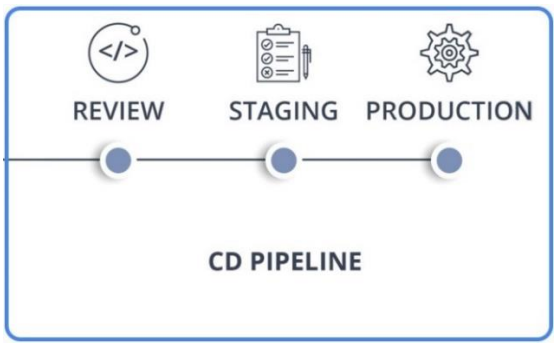

Fig. 2. A CI/CD Service

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.

3.2 Technologies

3.2.1 Jenkins

Jenkins is an open source Continuous Integration server capable of orchestrating a chain of actions that help to achieve the Continuous Integration process (and not only) in an automated fashion. It is a server-based application and requires a web server like Apache Tomcat. The reason Jenkins became so popular is that of its monitoring of repeated tasks which arise during the development of a project. For example, if your team is developing a project, Jenkins will continuously test your project builds and show you the errors in early stages of your development.

3.2.2 Docker

Docker is an open source platform to which uses the concept of containerization to provide Platform as a Service products. These containers run in a single operating system kernel and thus are more lightweight than virtual machines.

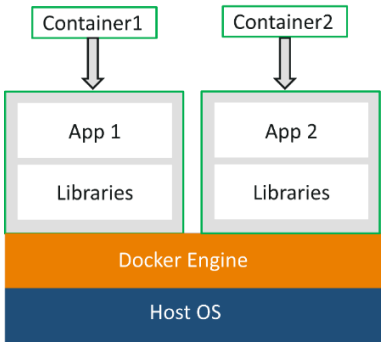

Fig. 3. A container is an abstraction consists of packages

A container is an abstraction consists of packages, code, dependencies of the application which makes them more stateless. Multiple containers can run on the same machine each running their own services and share the same OS kernel. They take up less space compared to the virtual machine.

3.2.3 Version Control

Each Component of the application when built need to have a version or a release tag so that they record the change set from the previous versions and the components can be rolled back / recalled later.

3.2.4 AWS

Amazon Web Services is a cloud services platform which offers wide range of distributed computing tools, storage, Networking & content delivery and infrastructure on demand.

3.2.5 AWS Lambda

This is a service provided by the AWS which runs the code on demand and requires no management of server. It is similar to REST API with an endpoint available to the outside world which when called, they run the desired operation they are programmed and returns the response.

3.2.6 AWS S3(Simple Storage Service)

This an object storage service provided by AWS that offers data high data availability, security and performance. It allows upload, downloading of files. It also provides REST APIs to the same operations.

3.2.7 Veracode

It is an application which provides security analysis such as static analysis, dynamic analysis which find the security flaws in the code of an application.

3.2.8 Mongo

Mongo DB is a database management system which uses document-based NoSQL database for high volume data storage. The database consists of collections where the records in the collection are called documents. It is called NoSQL since the data which is to be entered to inserted to the collection need not have a schema defined beforehand.

3.2.9 Node.js

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, npm, is the largest ecosystem of open source libraries in the world. The framework used in this application is the express framework.

3.2.10 The Express framework

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework:

- Allows to set up middleware to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

3.2.11 GitHub

GitHub is used for storing source-code of an application and track the complete history of all changes to the code. Each file has its own revision history that can be compared to previous changes, so that the new changes are identified easily and can be tracked. The projects are stored in repositories which contain many features viz. Forks and branches are copy of the existing repository which a developer can use it to modify without affecting the original source code. If any changes need to be done in the source code, the concept of pull requests comes into play, where the developer adds his code for a feature or fixes and raise it against the original source code, after the code is reviewed, the pull request is ‘merged’ to the parent repository. This technology comes very handy for integration of the CI pipeline, since version control is already there in GitHub as form of releases, commits. They can be checked out and the integration tests can be run against them.

4. AUTOMATION PIPELINE OF CI – CS – CD

4.1 Introduction

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

4.2 Theoretical Background

Most of the universities, clubs play a crucial role in organizing major events such as workshops, smart learning activities, fun activities, competitions etc. which is beneficial to the students in various ways. There are many online applications such as Facebook, twitter and many other social web applications that can be used for this kind of activity, but since they are global and is used along with their personal interests, there is no specific tool to manage the club such as organizing the meetings, schedules, events or other activities particularly. Although in application like Facebook is useful to the clubs, but since it is among various other global events, the club members can't organize the meeting, the role of each club member is not specified, the non-club students do not know who the organizers, the agendas cannot be discussed. The main objective of this project is to build a web application to ease the difficulties. A management tools for the clubs in University. They can also share the event they are created in the Events feed provided in the same application. As for the non-club members, they can't see an event unless they like or follow the club in social media applications like Facebook or follow a specific person in twitter. Hence to generalize these, this application is developed. Various club members have many personal club tasks that they have to do, a task manager is also provided in this application.

4.3 Aim of the Proposed Work

In this web application, both club/non-club members can use this web application and access the information about the clubs. The club members can discuss about the various agendas, hold meetings and can share their events globally. The non-club members can view the details of the events and register for the same. They will be notified by an email remainder about the events when registered. The clubs can be created by any of the current board members through an online registration form and they are verified by the admin of the web page through proof. The proof consists of the University ID of the registered member, they will be validated against the list of clubs along with its board members which is obtained from DSWD. When a member of the club signs up for the same, they are verified and added by any of the club member. Various roles are given to the members by the president and the club board members, they are able to create events, broadcast them in the event feed, can hold various meetings with the non-club members to discuss about the events or queries about the club, private meeting with the club members to discuss about the agendas of events or solve issues regarding the club. An Event Calendar has been made to keep the track of the events or other meeting that they have added in the calendar. There is also a timeline that keeps track of who joined the club, what events have been created and by whom, it also keeps track of what has been added in the club schedule may it be meeting, events or workshops. Since the non-club members cannot know about any events until its shared, A Bot has been provided to them in order to know about the events, they can interact with the bot, the bot is intelligent enough to fetch the information regarding the club or the events. The Bot is made public so that everyone including the club members can interact with the bot and get to know about the various existing clubs, their events and workshops. This application is developed using Node.js, using Mongo DB and Amazon AWS S3 as database and used to store other items such as images, proofs etc. The Bot has been trained using Google's Dialogflow, and calls are made to respective API when the user interacts with the bot. A Task manager for Club managers for their personal to do list is also provided in this web application.

4.4 Milestones and Deadlines

As the new features are added to the project, we implement the CI – CS – CD to deploy the changes in the live servers.

4.4.1 Objectives of the proposed work

- Design of the Club Soc Login system and the Club Dashboard.
- Design of the NoSQL Database in MongoDB.
- Constructing the Backend Database in mLab (Online MongoDB Storage).
- Design of Administrator Dashboard.
- Design and Implementation of the Club management tools such as events, workshop, member request.
- Design and Implementation of Event Calendar in front end.
- Design and implementation of task manager.
- Design and Implementation of public chats and Query section of Club.
- Design of private group chats
- Design and Implementation of Dialogflow Bot Framework.

4.5 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. In this 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. Below is the Full Pipeline of the process.

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, for example, the Controller for the Bot framework is called as `Chat Bot Framework`, similarly for the event management tool and the Messenger platform Tool, the controllers for them are termed as components in the automation.

4.6 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.

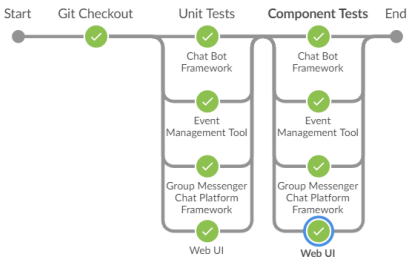

Fig. 4. CI Pipeline

4.7 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.

Fig. 5. CS Pipeline

4.6 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.

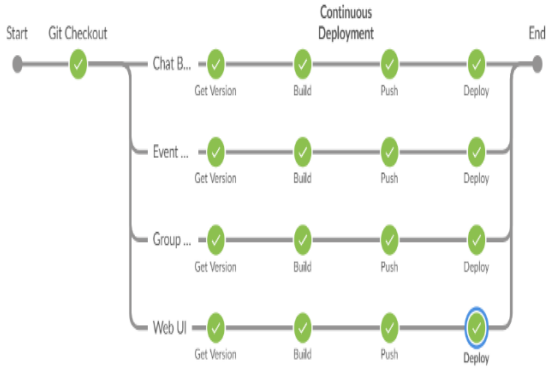

Fig. 6. CD Pipeline

4.7 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.

5. ANALYSIS OF IMPLEMENTING CI – CS – CD IN THE WEB APPLICATION

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. Improvements from the user feedback during the continuous development leads to usability improvements, the new requirements can be implemented on a daily basis. In this web application, 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.

6. 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. 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 analysis of implementing CICD in the web application 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. Operations goals are a key indicator to the products success.

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: https://stelligent.com/2016/04/05/continuous-security/ (2016).
[3] Shahin, Mojtaba; Ali Babara, Muhammad; Zhu, Liming. "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices". IEEE Access. (2017).
[4] 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).
[5] Fowler, Martin, Continuous Integration. martinfowler.com (2006).
[6] Samy Bengio, An Introduction to Statistical Machine Learning - Hidden Markov Models - https://bengio.abracadoudou.com/lectures/old/tex_hmm.pdf (2016).
[7] Pavel Senin, Symbolic Aggregate Approximation (SAX). https://jmotif.github.io/sax-vsm_site/morea/algorithm/SAX.html (2016).
[8] Deepak Raj D S is working as Software Engineer at Netskope. He is also an member of IEEE Computer Society and ACM. Deepak completed his B.Tech in VIT University Majoring Computer Science.
[9] Swarnalatha P is working as Associate Professor at VIT University. She belongs to School of Computer Sciences and Engineering.