International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2020 IJSRCSEIT | Volume 6 | Issue 3 | ISSN : 2456-3307
DOI : https://doi.org/10.32628/CSEIT2063166

Quality Maintenance and Monitoring using Azure CI pipeline and .Net Technologies

Ishwarya S[1], Dr. S. Kuzhalvaimozhi[2]
Department of Information Science and Engineering, National Institute of Engineering, Mysuru, Karnataka, India

ABSTRACT

The paper is about how the application is maintained and monitored using Azure CI pipeline. Maintaining and monitoring the quality of the software plays an important role in company's growth and performance. This is achieved using DevOps. Few years back agile methodology was playing a major role in the industry, software were deployed in monthly, quarterly or annual basis, which is time consuming. However, now industries are moving towards DevOps methodology where in the software deployed multiple times a day. This methodology provides the organization to constantly and reliably add new features and automatically deploy them across various platforms or environment in order to gain high performance and quality assurance products. Continuous integration and Continuous delivery/ Continuous deployment are the pillars of DevOps. Continuous integration, Continuous delivery and Continuous deployment are the continuous software development practices of industry. By automating the build, test and deployment of software, CI/CD bridges the space between development and operation teams. This paper also concentrates on how the Test Driven Development features of .Net technologies supports the quality maintenance and monitoring of the application.

Keywords: Continuous Integration, Continuous Delivery, Continuous Deployment, DevOps, Git, ASP.NET.

I. INTRODUCTION

With the competitive growth in the software industry, organization pays a significant attention in allocating resources for continuous practices, which supports in developing, deploying and delivering reliable and high quality products for the consumers. Previously, the organizations were maintaining the manual quality assurance (QA) teams, which represent the last security layer before any modifications could go live. Nevertheless, now due to the advent of continuous practices with the extension of agile with DevOps, there are teams for automation practices with the custom syntax for the quality assurance. The consumers of the products are always anticipating having continuous interaction with DevOps team so that they can furnish their continuous feedback. Devops is a combination of development and operations, which target the interaction of the software development and operation teams in software development lifecycle (SDLC). Devops Process or Lifecycle depends on the agility and automation. Each stage in the DevOps lifecycle focuses on breaking off the loop between development and operations process and driving production through continuous development, integration, testing, monitoring and feedback, delivery, and deployment to increase the performance of the organization and to provide a quality monitored and maintained product. The Devops lifecycle includes the seven main phases, the Figure 1 shows the different phases in the Devops Life Cycle and each phase is, described as below:

Plan – In this phase, the organization defines the business value and the requirements of the project. This phase may include tools like Jira or Git to find issues and for project management.

Code – This phase involves the designing of the software and the software code creation. Code creation includes the code Quality and performance.

Build – This phase involves managing the software builds and Versions .Automated tools used in this phase to compile, convert and package code for future release in production. The source code repositories or package repositories used for product release.

Test – This phase involves the continuous testing process, which includes both manual and automated tests. This testing phase ensures the ideal code quality.

Deploy – In this phase the coordinating, managing, scheduling and automating product releases into the production done.

Operate – This phase mainly involves in verifying that everything is flowing smoothly which supports in managing the software.

Monitor – This is the final phase of the Devops lifecycle where the identification and collection of information with respect to specific release and which provides the understanding the impact on end users.

The methodology presented in this work the developers after completing their part of code pushes it to the repository every time which is the common repository to all and hence the developers in different location can access the code which helps in providing the current status of the project to all in the team. Moving forward the task of build phase is to form a package by downloading the corresponding dependencies of the code so that there is no need for the test team to download the dependencies again. If errors occur in this phases it is sent to developers back again to fix. This is a manual process. This process automated by using the continuous integration process. This CI process monitors and checks for the new code arrival and builds it, if there is any new code arrived [5].

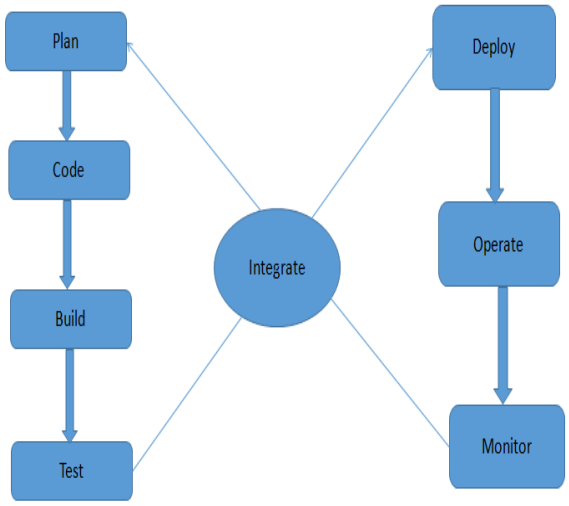

Figure 1 Devops Processes or Devops LifeCycle.

Agile enables the tactical and strategic framework of application software development, whereas DevOps focuses on both development and delivery segments. Combining Agile, lean and DevOps practices, the industry verticals increases the throughput and minimizes lead-times setting up the association, transparency, and assurance among Developers, QA and Operations as shown in Figure 2.

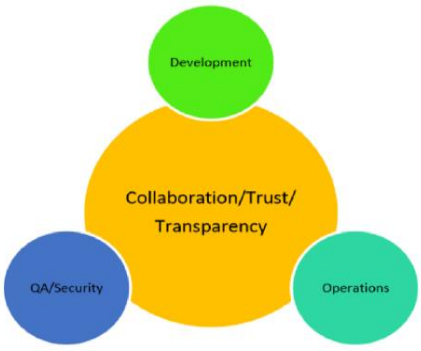

Figure 2 Devops Tactics for Quality Assurance

One of the main strategies of modern DevOps architecture is effective implementation of Test Driven development (TDD), and the tactical execution of TDD. Essentially, TDD environment enforces the test cases development prior to actual functional programming. The Asp.Net MVC framework supports this Test Driven development (TDD) [13].

II. METHODS AND MATERIAL

A. SYSTEM ARCHITECTURE

The DevOps includes the best practices, which consist of continuous integration and continuous delivery phases or processes. These practice are depicted as show in the Figure 3

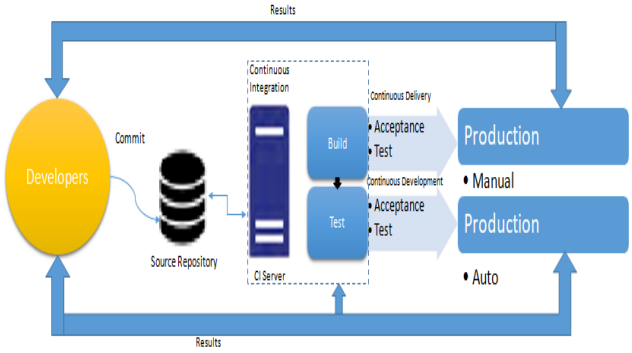

Figure 3 Relationship between the continuous practices.

The following describes the continuous processes involved in DevOps:

Continuous Integration (CI) – This is the software development wide development phase where the developers check in and merge their code to central repository multiple times in a day. After the merge the builds and tests runs occur.

This phase in DevOps mostly refers to the build and integration stage in the software development life cycle. It requires both cultural component and automation component. The basic challenges of implementing CI include more frequent commits to the common codebase, maintaining a single source code repository, automating builds, and automating testing. Additional challenges include testing in similar environments to production, providing visibility of the process to the team, and allowing developers to, easily obtain any version of the application. The main aim of this phase is to identify and look through the bugs much faster, reduces the time taken to validate, updates new releases to the software, improves productivity of the team and plays an important role in improving the quality of the software [12].

Continuous Delivery / Continuous Development (CD) – Continuous Delivery is the software development phase where the changes to the code done automatically built, tested and make ready for the production release. This extends the Continuous Integration by adding the code changes to the test environment, production environment or for both when the build is completed. This phase reduces the deployment risks. Continuous Delivery partially automated or fully automated.CD is either partially automated by using manual steps and it can be fully automated using workflow process. The continuous deployment and continuous delivery are different from each other where in the intended customers view or in the production environment. The aim of

the customers in continuous deployment practice is to, automatically and steadily deploy every change in the production environment. If the Continuous Delivery implemented properly, the developers will always have deployment-ready build artifact that has passed through a continuous process of standardized tests. The continuous deployment provides the continuous customer feedback loop in hand in the process of software production life cycle. There is a misconception that both continuous delivery and continuous deployment are same but these are two similar things but are different from each other. Continuous delivery is the extended process of continuous integration to ensure that the new changes released to customer fast in a sustainable way involves the automated release process along with the automated tests in turn where the application deployed at any point of time by clicking a button. Whereas, the continuous deployment is a step ahead of continuous delivery where deploys to the application takes place continuously and automatically to the customers or production environments. Human intervention is not present in this process and only the test fails will prevent new change to the production for deployment. This process the best approach to quicken the feedback loop with customers and remove pressure of the team since there is no release day anymore. Therefore, the developers can focus on building the software and they can see their work go live minutes after the work has been finished. One of the main benefits of continuous delivery process is to improved code quality since the bugs are identified, resolved early in the delivery process before they grow into larger problems later [7].

Continuous Testing - Continuous testing is like a backbone of CI/CD pipeline process. Continuous Testing is defined as a software testing process that involves a process of testing early, often, test everywhere, and automate. It is a strategy of maintaining quality of the code at each step of the Continuous Delivery Process. The aim of the Continuous Testing is test early and tests often. The focus of continuous testing is to achieve quality maintenance and monitoring. These testing supports in finding the risk address them and improve the quality of the product. This testing process accelerates the software delivery process. This involves the merging of siloed teams to meet modern enterprise needs. Connects or provides interaction between the development, testing, and operations teams. It maintains the same configuration for all similar or relevant tests. The other important benefit of the continuous testing is the speed. The delivery of the software becomes faster since there is a continuous testing process, which speed up delivery to production and release faster. The tests run parallel to increase test execution speed. Since, tests are automated the time spent on testing reduced and fastens the process. The test suites are prepared at every point when the code changes, merges or releases by these way tests can run at specific point rather than every test at once. This aids to minimize the time and effort on testing.

B. TOOLS *AND TECHNOLOGIES*

1. **NET technologies:** This is the new generation technology, which supports the application deployment on Microsoft Windows Server environment. It is one of the powerful frameworks, which supports in reducing time for developing complex web applications. .NET helps to develop high quality applications faster. .Net technology provides Quick and cost-effective development cycle. One of the main advantages of **ASP.NET** MVC is to support testability using TDD (Test Driven Development) and BDD (Behavior Driven Development). TDD provides the higher code quality and design. BDD is a testing process, derived from TDD, but mainly based on system behavior. BDD encourages collaboration in a

software project between developers, QA, project managers and the business team. This helps in maintaining and monitoring of the code quality in turn helps in improving the quality of the software.

2. **Git:** Git is a free, open source distributed version control system tool. This is a tool created to handle everything from small to very large projects with speed and efficiency. Git is a tool used to push code into one single central remote repository during software development lifecycle. This are used to monitor changes in the scripts and add them to the central repository. Developers push the codes changes made locally to central repository using the git commands which include git add, git commit and git push to add changes to the main repository.

3. **Jenkins**: In the continuous integration process, after the code is committed the build and test of the software done immediately. Jenkins is the tool that supports continuous integration and continuous delivery process for nearly almost all combination of the languages and source code repositories using pipelines and supports in automating other routine development tasks. This tool helps in determining the bugs or any other issues as soon as they introduced. This in turn helps in monitoring the quality of the product. Jenkins automates manual work of software development lifecycle [7].

4. **Docker:** Docker is a tool that uses container to which makes easy to create, deploy and run software applications. Docker Container is a standardized unit, which built to deploy a specific application or environment. Docker aids both developers and system administrator. Docker permits developer to write without concentrating about the system that it will be ultimately running. One of the main benefits of docker is that if once the package of an application and its dependencies are loaded to the container it will run in any environment. So the team can build a container, install different applications on to it and provide it to the QA team where they have to run only the container to replicate the environment [7].

III. RESULTS AND DISCUSSION

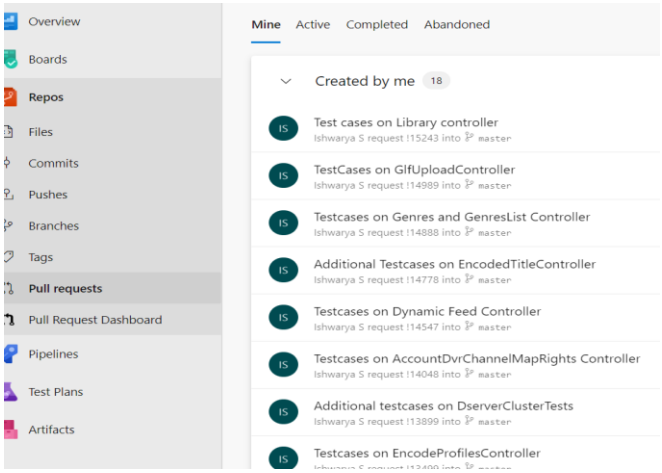

Figure 4. Create a pull request to push Code from local to central Repository (cloud) after executing git commands.

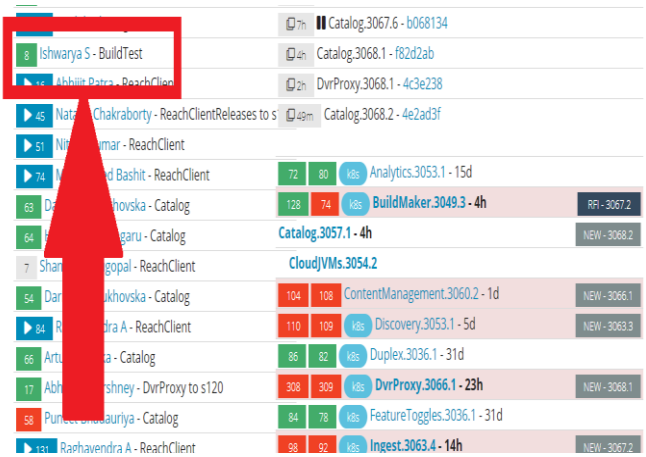

Figure 5. Build progress in pipeline.

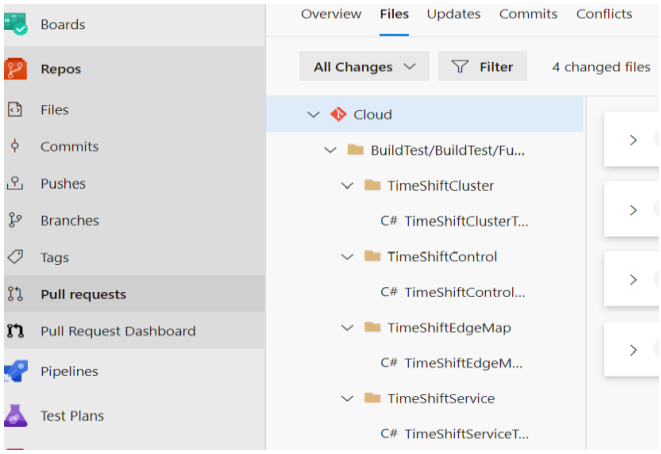

Figure 6 Files added to central repository after the build succeeded.

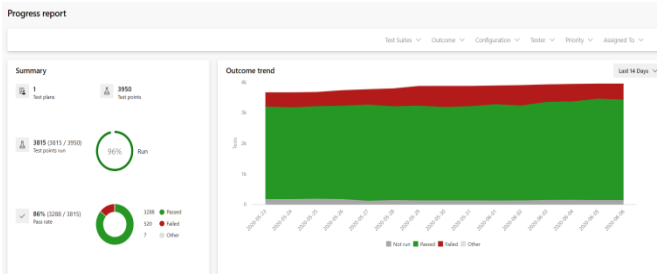

Figure 7. Progress report on the test suites running in pipeline.

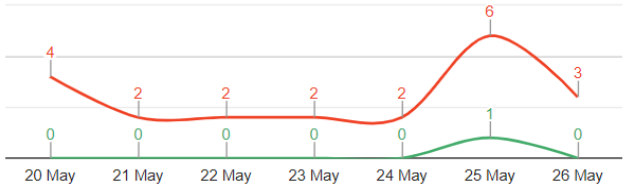

Figure 8 Graphical representation of weekly test runs.

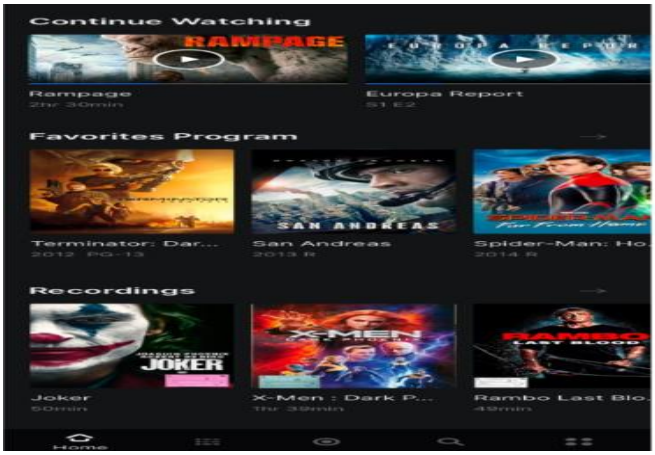

Figure 9 An overview on a Quality maintained and monitored product.

IV. CONCLUSION

In this paper, a discussion about how the quality of the software could be maintained and monitored using Azure CI/CD pipeline. The proposed methodology uses DevOps methodology where this provides the interaction between the developers and operational teams, which in turn helps to improve the quality of the software by obtaining the continuous feedback from the production environment or customers. The impact, of any release from Continuous testing correlated with data from operations and production, verifies the quality of the product. Continuous testing assures quality by simplified process by Continuous improvement at each stage of the software development lifecycle. In addition, this paper discusses how TDD (Test Driven Development) and BDD (Behavior Driven Development) of the ASP .Net framework from .Net technology helps to aid Continuous Testing provides, maintained and monitored software.

V. REFERENCES

[1]. Shahin, Muhammad Ali Babar, Liming Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices", IEEE 2016.

[2]. Prabal Mahanta, Anil Kumar Pole, Vittalraya Shenoy Adige, Rajkumar M, DevOps Culture and its impact on Cloud Delivery and Software Development, IEEE International Professional Communication Conference (IPCC), 2016.

[3]. Elisa Diel, Sabrina Marczak, Daniela S. Cruzes, "Communication Challenges and Strategies in Distributed DevOps", IEEE 11th International Conference on Global Software Engineering (ICGSE), 2016.

[4]. Hui Kang, Michael Le, Shu Tao, "Container and Microservice Driven Design for Cloud

Infrastructure DevOps", IEEE International Conference on Cloud Engineering (IC2E), 2016.

[5]. Matt Callanan, Alexandra Spillane, "DevOps: Making It Easy to Do the Right Thing", IEEE Software, 2016.

[6]. M Rajkumar, Anil Kumar Pole, Vittalraya Shenoy Adige, Prabal Mahanta, "DevOps culture and its impact on cloud delivery and software development", International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring), 2016.

[7]. Mojtaba Shahin1, Muhammad Ali Babar1, And Liming Zhu2, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices" IEEE Access (Volume: 5), 2017

[8]. Matej Arta, Tadej Borov, Elisabetta Di Nitto1, Michele Guerriero1, Damian Andrew Tamburri1, DevOps: Introducing Infrastructure-as-Code, IEEE/ACM 39th IEEE International Conference on Software Engineering Companion, 2017.

[9]. Zhenhua Li, Yun Zhang, Yunhao Liu, "Towards a full-stack devops environment (platform-as-a-service) for cloud-hosted applications", Tsinghua Science and Technology, 2017.

[10]. Wolfgang John, Guido Marchetto, Felician Nemeth, Pontus Skoldstrom, Rebecca Steinert, Catalin Meiros, Ioanna Papafili, Koastas Pentikousis, "Service Provider DevOps", IEEE Communications Magazine, 2017.

[11]. Len Bass, "The Software Architect and DevOps", IEEE SOFTWARE 2018.

[12]. Agarwal, Subhash Gupta, Tanupriya Choudhury, "Continuous and Integrated Software Development using DevOps", International Conference on Advances in Computing and Communication Engineering (ICACCE2018) Paris, France 22-23 June 2018.

[13]. DVSR Krishna Koilada, NetrixLLC, Business model innovation using modern DevOps, IEEE Technology & Engineering Management Conference (TEMSCON), 2019.

Cite this article as :

Ishwarya S, Dr. S. Kuzhalvaimozhi, "Quality Maintenance and Monitoring using Azure CI pipeline and .Net Technologies", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 6, Issue 3, pp.642-648, May-June-2020. Available at
doi : https://doi.org/10.32628/CSEIT2063166
Journal URL : http://ijsrcseit.com/CSEIT2063166