International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

Harare Institute of Technology
success through innovation

Innovation and Technopreneurial University

SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY

Master of Technology Degree Program

HIT 8000

Thesis

An Organisational Framework For Continuous Integration, Deployment and Testing in DevOps Environment

Academic Year 2022 – 2023

DEDICATION

Special dedication goes to my family, my wife Chipo and children Ryan, Reina and Ranielle. Also, special mention to my colleagues Nyasha Mutonhodza, Tafadzwa Manyetu and the 2022 MTech Cloud Computing Class. You can do it!

ACKNOWLEDGEMENTS

I am grateful to all those who I have ha the pleasure to work with during this another relate projects. Each and of the members of the dissertation committee has provide me extensive personnel and professional guidance. Special mention goes to my project supervisor Mr Mutandavari who taught me a great deal about scientific research. I wish also to acknowledge ECOCASH Holdings to explore the Cloud Infrastructure Devops

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

ABSTRACT

In the recent past, software release has been going through some major transformation. With software development becoming everything, most software developers are now trying to adapt to this high demand, and automation has become inevitable. Most organisations have adopted the use of pipelines of continuous Integration (CI) and continuous development (CD). This has been developed because of the over demanding for deployment and delivery of new software releases. This faster and continuous software delivery is made possible through DevOps. It uses the Agile methodology in which developers work together closely with infrastructure engineers to achieve the faster delivery, and deployment of applications more quickly and reliably. This Organisational Framework for Continuous Integration, Deployment and Testing in DevOps Environment overcomes the delivery challenges by improving the time to market, testing of release, availability. Based on experimental results, we demonstrate the advantages of for Continuous Integration, Deployment and Testing. This solution provides an effective and efficient way to generate, manage, customize, and automate Agile-based CI and CD framework through automated pipelines. This is demonstrate using a Kubernetes cluster using docker containers. The cluster works as a container orchestration tool. Rancher cluster manager is used to manage the cluster. Changing the principles of this solution and expanding it into multiple platforms (windows) will be addressed in a future discussion.

Key word: *CI/CD (continuous Integration/ continuous development), Kubernetes, Docker, Cluster, DevOps, Containers, rancher.*

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

Contents

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

LIST OF FIGURES

LIST OF ACRONYMS

CD	Continuous Development
CI	Continuous Integration

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

CHAPTER 1

INTRODUCTION

1.0 Introduction

The Continuous Integration, Deployment, and Testing framework's main objective has been characterised as the use of automated and computerised methods to transfer software between testing and production environments. Throughout the software delivery lifecycle, this process is both reliable and repeatable (SDLC). Among other things, the framework is anticipated to address software development issues such as average deployments per sprint, increasing the average number of successful deployments, reducing the average time per deployment, and mean time to execute repairs.

1.1 Background to the Study

The operation and operating environment of software companies have been greatly influenced by factors such as increased business competitiveness, rapid changes in technology, and market needs, resulting in the common challenge to enterprises of faster and more frequent software delivery (Amaradri and Nutalapati, 2016). Consumers' ever-changing needs impose an expectation for quick responses to their needs and requirements. Yang *et al.* (2020) discovered that software enterprises are responding with frequent releases, updates, and upgrades, as well as facilities for customers to provide continuous feedback, whereas previously they waited months for a product to release or act on customer feedback. Companies such as Ecocash, Amazon, Facebook, and others deploy software to customers on a regular basis, and by doing so, they increase their competitive advantage over their competitors. Due to both technical and non-technical issues, it is difficult to maintain a perfect business model while balancing technological change, and firms must be lean and agile across the whole software development lifecycle in order to overcome these difficulties. The need for automation procedures to be implemented grows tremendously as technology advances.

According to Wiedemann et al. (2020), agile software development practices have replaced waterfall software development practices over the past few decades, and this transition is unavoidably and unquestionably bringing businesses, operations, and software developers closer together in the direction of the goals they would have jointly established. In order to support this merger between operations and software developers, the software market introduced a sizable number of proprietary and open-source solutions. The cited proprietary and open-source solutions have since been developed and enhanced, which has resulted in the introduction of new processes including continuous integration (CI), continuous delivery (CDE), and continuous deployment (CD). In this research, a framework for continuous integration, deployment, and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

testing in a DevOps environment is being built and developed in order to assess and describe the existing status of CI/CDE/CD and establish a better framework.

1.1.1 Challenges with current deployment processes

The fundamental issue with manual deployments is that they are more error-prone and buggy than automated ones. The primary cause is because manual operations are vulnerable to human error:
Quick deployment. Copying and pasting deployment artifacts, as well as manually restarting services, are all manual processes (which sometimes failed). This has made the deployment procedure take longer.

• Additional undocumented server modifications made to accommodate particular code cause the configuration to deviate farther from the predefined state and the test environment. As a result, tests became unreliable and allowed defects to pass because of variations between the environments used for testing and production.

• Rewind procedure: As with deployment, the process on its own takes time, which results in more downtime, affecting the mean time to recover.

• Human Influence. Similar to how automated systems can duplicate things precisely, manual operations cannot. Human errors occur despite the finest efforts and greatest caution.

1.2 Project Rationale

1.2.1 Challenges for Development Team:

Developers are motivated and eager to adopt new strategies and technology to address the problems facing organisations. They do, however, confront a number of difficulties, such as the following:

• There is a lot of pressure for deliveries to be made on time due to the competitive market.

• They also need to support the implementation of new capabilities and production-ready code management.

• Because the release cycle is lengthy, the development team must make a number of assumptions before deploying the programme. It takes longer in this situation to fix any problems that came up during deployment in the production or staging environment.

1.2.2 Challenges for Operations Team:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

As they search for stability, the operations staff is constantly cautious when changing any resources or utilising any new technologies or approaches. They do, however, confront a number of difficulties, such as the following:

• Resource contention: managing rising resource demands is highly challenging

• Modification or redesign: This is necessary for the programme to run in a production setting.

• Diagnosing and Resolving: Following the separation of application deployment, they must identify and fix production-related problems.

1.3 DevOps Overview

Instead of delivering a large number of application features, businesses are attempting to test whether a small number of features can be rolled-out to their customers through a series of release iterations. Due of its many advantages, such as greater software quality and quick user feedback, this encourages high customer satisfaction. The following requirements must be met by businesses in order to achieve these objectives:

- Increased deployment frequency,
- a shorter lead time between fixes,
- and a faster mean time to recovery in the event that a new release crashes the application

All of these goals are expected to be accomplished by the DevOps CI/C system, which also aids with seamless delivery. DevOps has been used by local companies including Old Mutual, PMI, and Econet Wireless to attain performance levels that were unimaginable just a few years ago. They perform dozens of deployments each day while providing the highest levels of security, stability, and dependability.

DevOps is a framework for processes that ensures collaboration between the development and operations teams so that code may be sent to production environments more quickly in an automated and repeatable manner. The phrases "development" and "operations" are combined to form the term "DevOps." The pace at which apps and services are delivered can be increased with the aid of DevOps. It enables businesses to provide effective customer service and boost their marketability. DevOps can be summed up as an improvement in communication and collaboration between development and IT operations.

1.4 Aims

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

This study's primary goal was to clarify how organizational framework for continuous integration, deployment, and testing in a DevOps environment improves software delivery. It sought to address the financial (cost-serving), internal, and innovative perspectives of the consumer.

1.5 Objectives

1. To come-up with continuous integration of DevOps tools that will enhance system performance.
2. To automate deployment in order to eliminate system bugs, and human interference.
3. To test the deployment system using the container orchestration tools and Kubernetes clusters

1.6 Scope of the project

The scope of this process is delimited to the following:

- Integrating DevOps tools (GIT, Maven, Docker, Kubernetes) to come up with a deployment automation process.
- Management of all deployment through a central management portal (ranger)
- Create a central repository to keep track of all deployments in case of rollbacks

1.7 Chapter Summary

This chapter introduced the problem and how it is manifesting in the current systems and the proposed design to innovate and circumvent some of the current challenges in continuous development, deployment and testing in a DevOps environment. The challenges with the current deployment methods and strategies were exhumed and exposed which this project aims to improve and make the process smoother, effective and efficient to encourage innovation. The next chapter reviews literature related to DevOps, development, deployment and system testing.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

CHAPTER 2

LITERATURE REVIEW

2.0. Introduction

This chapter is a summary of earlier research that is pertinent to this topic. A brief summary of the general research on strategic business-IT alignment is provided in this section, along with discussion of research gaps on operational alignment and misalignment that point to the need for new theories, models, and frameworks. The section also provides the foundation for a comparison of other continuous integration, deployment, and testing methods for software development and the DevOps environment.

2.1 Theoretical framework

According to Creswell and Creswell (2018) theoretical framework consists of concepts, together with their definitions, and existing theory/theories that are used for a particular study. The theoretical framework must demonstrate an understanding of theories and concepts that are relevant to the phenomena under study. This study will be guided by the DevOps Maturity Model (DMM) propounded at Capgemini by Menzel and Macaulay (2015)

2.1.1 DevOps Maturity Model (DMM)

It is impossible to implement DevOps with a single project. Organizations must deploy tools and procedures in a coordinated programme and gradually change their culture. By determining the stage that the organisations have reached and then developing a road map to get there, the progress must be monitored. The DevOps maturity model (DMM), developed by Menzel and Macaulay in 2015, is a framework that enables enterprises to determine their present maturity level and the steps needed to advance. DMM uses the fundamental characteristics of people, process, and tools to evaluate maturity. This model can also be utilised as a road map for development inside a project, department, or organisation as a whole. The DMM can be used as a reference to describe the existing DevOps maturity in an organisation and aids stakeholders in understanding and developing an implementation plan. The DMM has five stages of maturity, from Level 1 with a siloed team, manual processes, and ad hoc decisions to Level 5 with a highly matured "One Team" and fully automated, dynamic procedures.

2.2 Conceptual framework

According to Gokarna and Singh (2021) DevOps is a set of procedures made from the combination of Development and Operations meaning DevOps needs a set of tools to ensure the performance of the function

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

of combination and integration. DevOps, then, is a single and dedicated software development team that is responsible for the management of software development, testing, and operations while giving an assurance that the entire product cycle runs without interruption. According to Ebert *et al.* (2016), DevOps has four Dimensions which are collaboration, automation, measurement and monitoring.

According to Donca *et al.* (2022) DevOps is the extension of the agile method of software development that focuses on the continuous delivery of the software along with continuous integration. There is consensus that DevOps improves collaboration and communication and also offers fast and continuous delivery, regular updates, increases reliability (Riti, 2018). Hochbergs and Sjödahl (2020) explained that DevOps is not necessarily the same process, or set of processes, in every development context, and misunderstandings occur when discussing the subject, due to different personal experiences, understanding, and views or opinions. The main aspects of DevOps have been noted in literature included the organisational culture, automation, measurement and sharing. Culture implying the "people over processes and tools" belief taking the people an integral and critical component of the whole system. Wiedemann *et al.* (2020) recommended that organisations should analyse the current workflows and target those areas which have a scope for optimization.

Continuous Integration (CI) is a well-known software development technique that is being utilised extensively throughout the world. CI is a process where members of the development team often integrate and merge their development work, possibly several times daily or once per week. According to Wiedemann *et al.* (2020) continuous integration is a software development practice in which developers routinely combine software code changes in a central repository, after which build, testing, and launch are automatically performed and seamless that users may not even notice. CI enables software companies to have quick response to changes, challenges and issues, shorter and frequent release cycle, improve software quality, and guarantees an increase their teams productivity (Helsinki and Wikström, 2019). Continuous integration entails the development and testing of automated software.

2.2.1 Continuous Deployment

This process is according to Bobrovskis and Jurenoks (2018) closely related to continuous integration and makes reference to the keeping of applications deployable at any point. Continuous deployment includes the involvement of frequent, automatically done deployment of the master branch to a production environment after automated testing would have been completed.

2.2.2 Continuous Delivery

The discipline of being able to consistently release application/code changes at any time is referred to as continuous delivery (CD) in DevOps. Along with passing automated tests, the application must also have all the configurations required to launch it into production. Continuous delivery in DevOps refers to the entire development lifecycle, from inspiration to build to readiness to delivery to production.

2.2.3 Continuous testing

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

In DevOps, continuous testing involves testing at every stage of the development life cycle. One of the major objectives of the continuous testing stage of DevOps is to evaluate the quality of the software being designed as part of a continuous delivery process, by testing early and proceed with the same process more often to fish out bugs and other errors (Shahin, Ali Babar and Zhu, 2017). Traditional manual testing involved software being handed off by one team to another at various stages of the development lifecycle instead of it being embedded in the automated pipeline. In a continuous DevOps process, a software change, also known as a release, continually moves from development to testing to deployment.

2.2.4 Quality Assurance

Quality assurance (QA) in the software development lifecycle refers to the process of achieving or maintaining a desired level of quality in a service or product (Yang *et al.*, 2020). The major aim of quality assurance is to deliver consistent results by using agreed standard processes and procedures, processes and includes all activities centred around ensuring software meets certain requirements before being released.

Software development firms or companies use continuous integration at the building or integration stage or level of releasing the software. The stage includes not only the automation but also the development culture component. According to Gokarna and Singh (2021), the main objective of the continuous integration (CI) process is to quickly identify and fix or correct software errors, to improve the quality of the software and to reduce the overall time spent on checking and releasing new software updates

2.3 DevOps in the Real World

Wiedemann *et al.* (2020) observed that in real life, the DevOps hybrid method integrates the various software development tasks and blurs the distinctions at every stage. Using an experimental, multiple case study technique based on some interviews with DevOps experts, it is shown that the planning, development, testing, and operation of software product activities inside a collaborative cross-functional team within the IT function is viable. These DevOps teams jointly plan, develop and operate IT software and architecture solutions using an integrated software delivery lifecycle and thereby bridging development and operations (Wiedemann *et al.*, 2020; Gokarna and Singh, 2021). There is consensus in literature that DevOps integrates the advantages of agile software development to react quickly to customer demands and also broadens agility to operations such as software architecture, responsibilities and knowledge (Shahin, Ali Babar and Zhu, 2017; Gokarna and Singh, 2021).

The DevOps method extends agile software development by focusing not only on the development subunit, but also on the operations subunit adding speed of delivery and bridging the gap between the two silo IT subunits to form a cross-functional team (Shahin, Ali Babar and Zhu, 2017; Donca *et al.*, 2022). By integrating or combining the viewpoints of software engineers and software operators into one cross-functional team,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

businesses hope to achieve internal IT alignment, foster consensus and fluency within the heterogeneous cross-functional teams, and increase levels of agility. Ebert *et al.* (2016) believed that by combining the activities of development and operations into single cross-functional teams, the DevOps methodology enhances alignment between the two distinct subunits.

Observations by Wiedemann *et al.* (2020) report that DevOps projects face four major related challenges:

- Deliberate implementation of a purpose-built development and production environment derived from legacy application life-cycle management or product life-cycle management environments;
- Breaking down and simplifying complex software architectures and feature sets into manageable sets that can be produced and deployed independently;
- The maintenance of a configuration and build environment that provides constant visibility of what's deployed, with which versions and dependencies;
- and bridging the traditional cult of deployment.

2.3.1 The DevOps Reference Architecture

According to Sharma and Coyne (2015), a reference architecture refers to a framework that provides a template of solution that has already been proven to work by utilising preferred methods and capabilities. The DevOps reference architectures help practitioners access and use the frameworks with guidelines and other material that they may possibly need to design or architect a platform that accommodates all stakeholders including people, processes, and technologies (Sharma and Coyne, 2015).

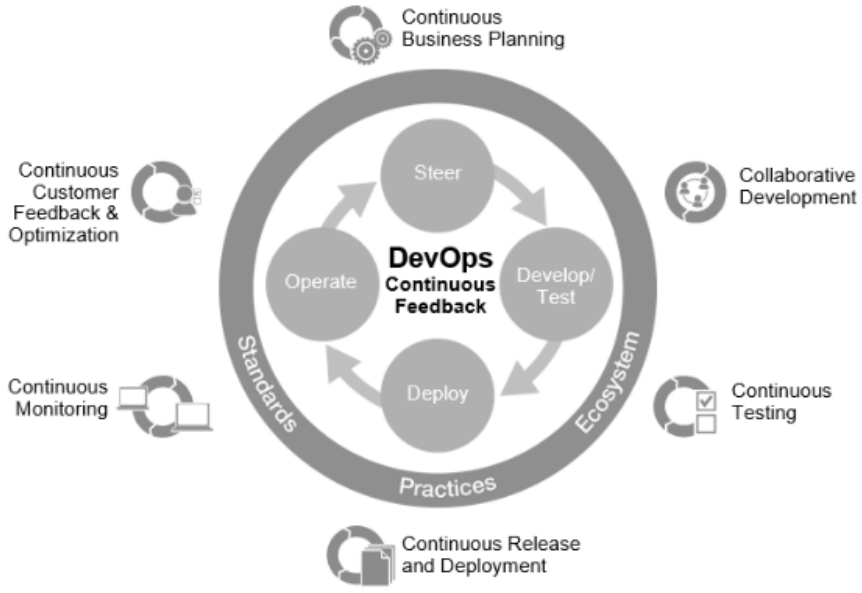


Figure 1: The DevOps Reference Architecture

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

Professionals in the operations and development departments frequently differ, which makes the DevOps environment difficult (Chandramouli, 2022). While operations experts want to maintain the infrastructure's stability, software developers want to release upgrades more regularly. The ultimate aim of DevOps, according to Chandramouli (2022), is to create a setting where developers and operators collaborate to continually improve operations by bridging traditional operational and developmental lanes. In order to ensure that customers continue to receive services with greater standards of quality and more features in a way that fits their needs, the overall DevOps strategic purpose is to increase return on investment. Transitioning toward DevOps is practically more challenging with legacy software, and some case studies involving application development and Web environments reported easy migration to DevOps because of a single source and thus orchestrated rollback was feasible (Wiedemann *et al.*, 2020; Gokarna and Singh, 2021). Henning and Hasselbring (2021) recommended that when embracing a DevOps culture, developers must take a full stack developer approach, in which they take responsibility for the testing and release environment. Adding to that, development teams are anticipated to have extended skills that go beyond just code knowledge but inclusive of database administration as well as testing. Furthermore, as the functional silos' boundaries are fading away, more intense collaboration with other team members is required, for example, testers can be paired with developers to ensure that the development team performs test-driven development and CI (Luz, Pinto and Bonifácio, 2019). When testers and developers are paired, both groups can gain technical knowledge about the activities and expectations of the other and contributing to the quality assurance team that gives an assurance that the automation of all cases to be tested are assured to meet quality expectations as far as full code coverage is concerned. From the perspective of the operations team, DevOps greatly affects culture and discipline because operations teams are expected to continuously connect to other functions without losing control meaning the IT operations and development teams must closely collaborate to achieve the continuous process promoted by the DevOps team (Wiedemann *et al.*, 2020). Additional, infrastructure monitoring and application performance management are more important, and communication among team members must be fluid (Wiedemann *et al.*, 2020; Gokarna and Singh, 2021).

As it was noted that DevOps lacked the discipline that engineers are accustomed to, Farroha and Farroha (2014) developed a framework for guaranteeing that overall enterprise performance and security are not compromised while embracing DevOps. For the integration of software components in a European research project, Bobrovskis and Jurenoks (2018) describe a DevOps approach. They discovered that the DevOps workflow had made the development teams more adaptable to changes and capable of operating independently, resulting in updates being done more frequently. They assert that the strategy decreased communication burden as well as speeding up delivery, but they also acknowledged that it was difficult to fully quantify the results because they had only recently begun gathering measurements. Rejström (2016)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

identified the Knowledge, Skills and Abilities (KSA) needed for both developers and operations personnel to support the main aspects of DevOps: Culture, Automation, Measurement and Sharing.

DevOps improves the performance of the development, operations, as well as the quality assurance teams. Rejström (2016) and Riti (2018) and have identified the main aspects of DevOps as Collaboration, Automation, Measurement and Monitoring, and also developed a framework to provide an understanding of how DevOps works.

Automated DevOps models have been established to help organisations understand their duties and the DevOps infrastructure, and the majority of them can increase communication and knowledge transfer (Bobrovskis and Jurenoks, 2018). Additionally, according to Bobrovskis and Jurenoks (2018), there is no one way that DevOps should be carried out by different businesses because this is where they would focus their competitive advantages while adhering to a variety of standardised principles, practises, and procedures. In order to understand the problems related to operations, Donca et al. (2022) advised that in addition to good communication with the operations teams, developers should look into several other aspects such as standards, organisational process descriptions, and studies about different types of operations failures.

2.4 Automation in DevOps

According to Bheri, Vummenthala, and Molleri (2019), automation enables developers, operators, testers, and other DevOps stakeholders to automate the processes involved in the development and deployment of software. Automation lowers risks and delays since human completion of tasks like integration, testing, deployment builds, and software delivery takes time and is prone to error (Bobrovskis and Jurenoks, 2018; Hornbeek, 2018). Delivering a new release, especially in a DevOps context, may be challenging work for many applications. For web applications, this may entail setting up and configuring web servers as well as correcting any foreseeable or unforeseen faults so that the new version can function as intended. According to Chandramouli (2022), these tasks are challenging to complete manually because they may be prone to mistakes, cause delays, and incur additional costs. Most of these issues affect the operations teams increasing the probability of their conflict with the development team. The deployment pipeline, has been clearly defined by Shahin, Ali Babar, and Zhu (2017) as the comprehensive and clearly defined procedure that changes in the source code are procedurally anticipated to pass through so that it become visible to the final users, is one of the basic concepts in DevOps automation. The process of moving software that is being development from the stage of version control to that of the production environment is normally referred to as the development pipeline. An illustration of a development pipeline workflow may be found below.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

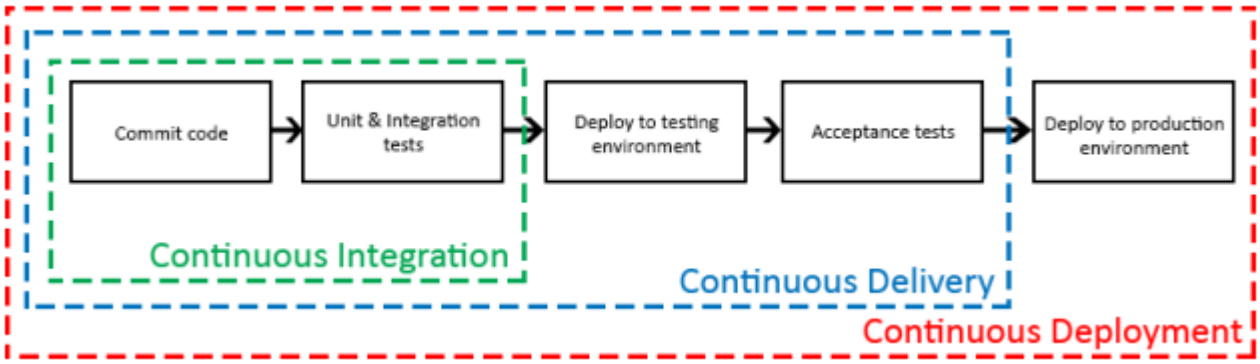


Figure 2: CI/CD pipeline workflow (Truong and Klein, 2020)

The processes are automated using a deployment pipeline, and any changes to the system must be verified by running them through a set of automated tests in the pipeline. The deployment of the changes to the software being developed to the testing, staging, and finally to the production environments will be possible if the changes are successful. The success of DevOps is founded on four outlined collaboration principles: mutual respect, dedication to shared objectives, collective ownership, and shared values (Hornbeek, 2018; Bheri, Vummenthala and Molleri, 2019). All team members must appreciate each other's values, work well together, be dedicated to the project's success, and be on the same page in order for DevOps to be successful. Teams need to share ownership in order to guarantee the high quality of the software they build and that their customers will be satisfied. Any project type, including major projects, small projects, and mobile application projects, can use the DevOps principles. The following have been highlighted as DevOps practises or enabling techniques: continuous planning, Continuous integration, Continuous deployment, Continuous testing, Continuous delivery, Continuous monitoring and Continuous feedback from the teams and the customers (Wahaballa *et al.*, 2015; Donca *et al.*, 2022).

2.5 Benefits of adopting DevOps in Organisations

There is a wealth of material on the advantages of implementing DevOps in organisations. In a DevOps context, teams were able to reach project milestones with more speed and efficiency, according to Donca *et al.* (2022) who noted a qualitative difference in how teams operate in a manual system. Rejström (2016) noted a notable reduction in the typical release cycle time as well as beneficial effects on the operations and development teams' performance, which were corroborated by the quality assurance team. DevOps has a favourable effect on web service development by fostering greater trust and teamwork, which enhances work flow, according to Donca et al. (2022).

The primary studies have identified the following advantages: enhanced cooperation, trust, and communication between development and operations teams; enhanced software deployment quality; increased responsiveness to customer needs; increased reliability; enhanced code quality; streamlined implementation

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

because teams work independently; and decreased costs (Ebert *et al.*, 2016). DevOps aa a well-structured system is supported by observable structures and agreed standards, which makes it easier to realise the beneficial structures and standards for development and operations. It also bridges the gap between development and operations teams, lessens fear of change and risky deployments, accelerates time to market, allows for continuous feedback, balances costs and quality, improves predictability, and boosts the efficiency of the entire organisation (Gebril, 2020).

2.6 Current DevOps practices and tools

The term "source code management" (SCM) was first used by Bheri, Vummenthala and Molleri (2019), who described it as a collection of procedures for monitoring changes to software's source code. Versioning and remote teamwork are made possible by SCM since it allows the software developers as well as the other team members to share code and work from different locations. Version control fundamentally regulates communication between developers during the coding phase of the lifecycle as well as changes to the source code. SCM can be used to monitor changes in any emerging artefact in the software development process, and GitHub is the most widely used and well-liked version control solution since it supports distributed systems, is freemium, open source, and other popular sources code tools include Subversion, Mercurial, and Bitbucket (Chandramouli, 2022).

According to Bheri, Vummenthala and Molleri (2019), build is the process of preparing an executable program or set of programs out of a set of source code files for a particular software product which can involve the compilation of the source code into machine instructions for a specific computer architecture, but it can involve other steps such as handling dependencies. Examples of Build Tools include Ant, Maven, Gradle, MS Build, NANT, with Gradle being the popular due to its combination of the favourable features from other tools.

Using CI tools, developers can automatically integrate and also merge code that is provided to the shared source-code of the system or software that is being developed for building and testing. Major examples of CI tools are Travis CI, Jenkins, TeamCity, and Codeship, which provide pertinent feedback right away to notify developers if something went wrong. By monitoring changes to any artefacts used and managing the various versions of each artefact, configuration management is the DevOps process involving the establishment and maintenance of consistency of software products and artefacts throughout the software development life cycle. Examples of configuration management tools include Chef, Puppet, and Ansible (Bheri, Vummenthala and Molleri, 2019).

Microsoft Azure and IBM services are two prominent examples of cloud platforms that incorporate collaboration and deployment in support of DevOps processes.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

Automated testing is a crucial procedure because it enhances teamwork and the calibre of the software output. The major tools for automated testing are Cucumber, Selenium, and JMeter, and DevOps constantly does testing in simultaneously with automation.

Containers are frequently used for platform development and application deployment in infrastructure, shortening the delay between creating code and production. Containers are simple to deploy and maintain for DevOps personnel, which reduces overhead. Micro-services are operated in separate settings; therefore, containers help them to deploy independently.

Deployment tools: These tools are designed to handle the automatic deployment of new releases. Without the need for human effort, tests are run on the software after each change that would have been made in the source code, and fresh updated versions are issued. Some examples of deployment tools are Capistrano, Jenkins, and Ansible.

Monitoring tools: utilising monitoring techniques, infrastructure issues that can have an impact on business solutions are identified and attempted to be fixed. The cloud applications benefit more from monitoring tools, of which prominent examples include Nagios NewRelic, Graphite, and Cacti.

As DevOps is primarily founded and reliant on trust among stakeholders, open and comprehensible communications, as well as effective cooperation, it encourages the various teams involved at all stages to share responsibility, ideas, and goals of the whole process. The main application of collaboration tools like Jira and Slack is in DevOps (Bheri, Vummenthala and Molleri, 2019).

2.7 Challenges of Adopting DevOps in Organisations

Continuous integration and deployment in organisations is currently facing multiple challenges while adopting DevOps. Some of the challenges that occur while adopting DevOps in organisations include: organisations may lack proper management structures; employees may lack proper training for DevOps ; so the unclear definition and goals of DevOps adoption; the geographic location spread of the development and operations teams; developers may be afraid that DevOps will overburden them with other responsibilities such as operations, and some team members may not be open to the change due to fear of not having in-depth expertise in both areas; individuals may be interested in only their area of expertise; the differences between the development, testing and production environments may become a complication in the collaboration and also continuous delivery and deployment (Wahaballa *et al.*, 2015; Gokarna and Singh, 2021; Scott, 2021).

The cautious adoption of DevOps was a bottom-up process, meaning it was recommended by practitioners to management, in the organisation where their research was conducted; however, the operations teams' resistance, the upkeep of legacy systems, a notable absence of a proper management structure, an absence of support from the executive management team, legacy systems, hardware dependence, and compatibility issues

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

may cause delays in the release of new features to production. Shahin, Ali Babar and Zhu, (2017) also noted the absence of automation tools needed to regularly deploy new features in the field of embedded systems. Additionally, it was highlighted that certain businesses were able to get system performance data, which aids in ongoing product improvement. As a result, the absence of data on the usage of features was noted as a barrier to the adoption DevOps in organisations.

Infrastructure, organisational structure, development strategy, and ongoing operation team involvement all need to alter. Addition to the adoption process, the organisation is anticipated to adhere to agreed principles and practises and include certain practises into its development processes. The facilitation of communication as well as coordination among the development and operations departments is one of the principles and practises that must be followed in the organisation to achieve synergy between the two departments. To support technical practises, the organisation must adhere to the concepts of automation, sharing, and communication. The following DevOps practises have been discovered in the literature: Continuous testing, monitoring, and feedback are also included in continuous planning, continuous integration, continuous deployment, and continuous delivery (Wahaballa *et al.*, 2015; Shahin, Ali Babar and Zhu, 2017; Chandramouli, 2022). These processes must be automated and clearly documented and monitored.

2.8 Chapter summary

Related literature was reviewed from various information sources and formats to understand the DevOps operating environment, current tools, processes, procedures and challenges. Current frameworks were also analysed to guide the crafting of a new improved framework.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This chapter delves into the methodology used in the data collection, analysis, and interpretation for the study on the framework for the development, deployment and testing systems under DevOps environment. Polit and Hungler (2014), defined research methodologies as various methods of gathering data, processing it by examining and analysing it with the goal of obtaining results for a research study. Leedy (2016) Methodology is defined as an operational system in which facts are organized so that their significance can be understood more clearly. The methodology employed attempts to provide an analysis of the research plan and research tools to be used in this study. It provides information on the research population and sample. This chapter will also include information on the sample sizes used and the responses.

3.2 The research methodology

The researcher employed a mixed methods approach to analyse the current systems and challenges at Ecocash Holdings. Generally, there are three types of research methods: Qualitative, Quantitative and Mixed methods. One major difference between the qualitative and quantitative methods is that qualitative methods collect qualitative information that is hard to quantify, while quantitative methods use finite numbers and quantities (Bowen, 2011; Creswell and Creswell, 2018). The qualitative methods are flexible in nature, and they involve open-ended questions to extract in depth information. The mixed methods approach combines both qualitative and quantitative data hence is particularly useful when a researcher wants to gain a detailed understanding about a phenomenon as in this study. The motivation behind choosing this approach was to be able to get more reliable and valid results than using either a qualitative or quantitative approach separately (Creswell and Creswell, 2018). The research aimed at identifying all possible challenges that the organisation was facing in their current adoption of DevOps, during both the initial and later stages of adoption, with the intention of providing solutions to overcome these challenges with a new proposed framework.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

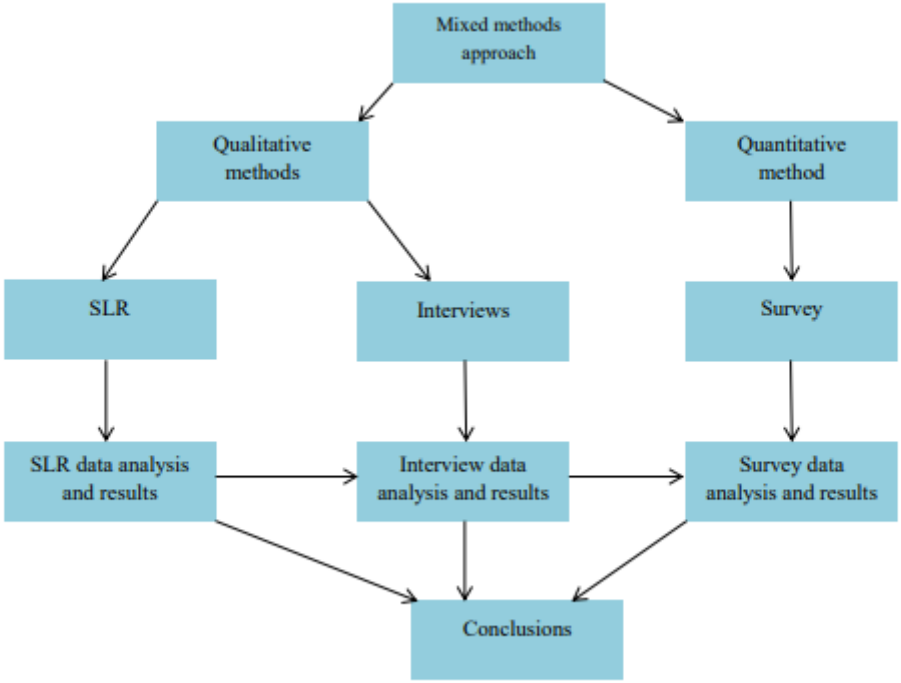


Figure 3.3: The mixed methods approach design (Amaradri and Nutalapati, 2016)

3.3 The design science Research process

The DSRM is made up of six distinct processes that may overlap: identify the problem and motivate, define objectives of a solution, design and development, demonstration, evaluation, and the communication of the developed performance system. Identify the problem and motivate: in this activity, the specific research problem is defined.

Define a solution's objectives: the second action of the DSRM involves determining a solution's goals. These goals are established in accordance with the problem definition. Understanding the condition of problems and the literature's existing remedies is necessary for the definition of objectives. As stated, chapter 2 conducted a thorough and in-depth literature research. Design, development, and validation: Produce the artifact, which could be an algorithm, a conceptual framework, a technique, or a combination of these. Any created object in which the research input is incorporated into the design might be considered a design research artefact (Al-Ghamdi, 2019). The researcher incorporated experts who indirectly participated in the process to verify that the design is appropriate and would function as intended. Demonstration: In this activity, the artifact is utilised to show how it resolves the challenges and issues raised in the DSRM's first activity. There are various ways to carry out this demonstration and in this project, a case study was used of a specific company as proof or experimentation. The system used in this investigation at the aforementioned was Ecocash Holdings.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

The links between the research, the research environment, and the knowledge base of the research in the design science approach is informed by the evaluated literature and form the framework for design science research. The operating environment was built on these three components. The goal was to develop an artifact that would be expected to perform a certain purpose in the application domain (in this case, Ecocash), using the knowledge and information from the knowledge base, which is typically informed by theories, standard procedures and methodologies.

The framework for the research process that served as the basis for the system's design is shown in the simplified diagrammatic format below.

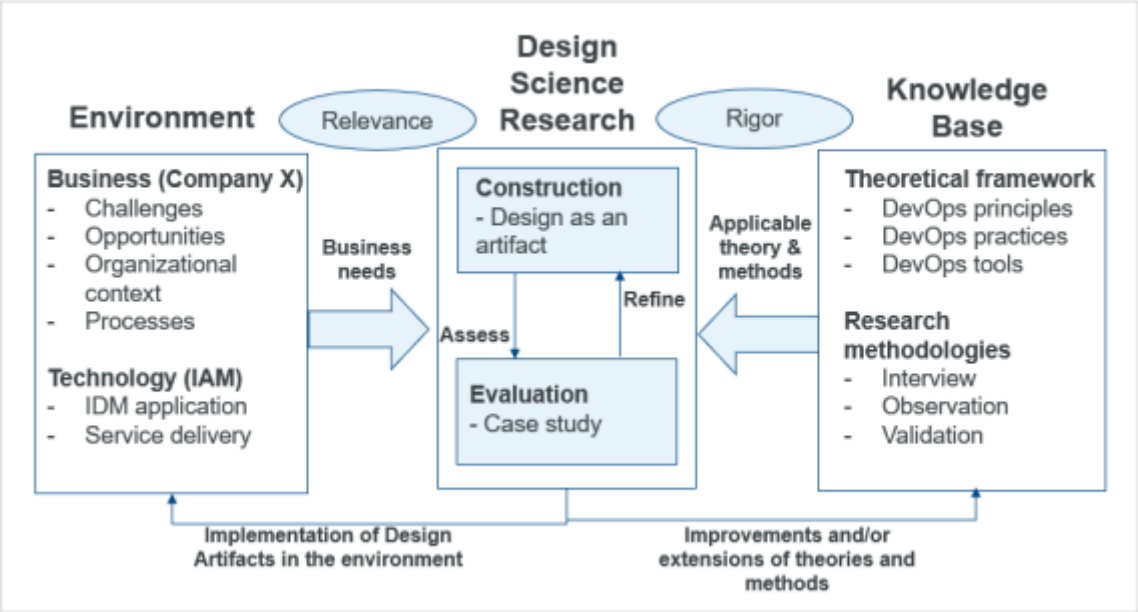


Figure 3.4: Design Science Research Model (adopted from Blomberg (2015))

The diagram above shows the 3 main components of the process, the Environment which was the case being studied to aid in the deployment, the designing process being the data gathering, experimentations, trials and errors in finding a suitable solution and the knowledge base to guide both in terms of the research process and the designing and guiding behaviour as the researcher interacted with the environment.

To understand the environment, data was gathered to understand (Company X), in this case Ecocash as the case study.

3.4 Research Population

The targeted population were employees of Ecocash working in the system development, testing, deployment and operations unit.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

3.4.1 Sampling and Selection of research participants

To select the participants of the research, a non-probability sampling technique, the purposive or judgemental sampling method was employed. The participants were supposed to be key informants in the various capacities of development, testing and operations units at Ecocash (Kozludzhova, 2020).

3.5 Data Collection Instruments

According to Kozludzhova (2020), data collection instruments are the tools used in the extraction of data, information from the knowledge, opinions and views of the research participants or respondents selected to participate in the study. The researcher utilised various instruments in the collection of data to meet the objectives of this study. The researcher administered structured interviews to the key informants and other selected participants for the study, also practices document analysis as they analysed documents with standard operating procedures, hierarchical structure and other workflows that would inform the new framework. According to Creswell and Creswell (2018) no single instrument may be considered to be adequate in itself in collecting valid and reliable data. Therefore, to obtain adequate and reliable information for this study multiple data gathering instruments were utilised in a method called triangulation. Through triangulation the researcher used a combination of different data collection techniques namely the structured interview and the focus group interviews.

3.5.1 Document analysis

Bowen (2011) defined document analysis as a systematic procedure for reviewing or evaluating documents. These document maybe in any format including both printed and electronic documents. Document analysis requires that data contained in those documents be examined and interpreted in order to elicit meaning, gain understanding, and develop empirical knowledge (Leedy and Ormrod, 2005; Bowen, 2011). Documents contain text and images that have been recorded without a researcher's intervention but helping in observing such important data as the structure of the organisation, processes and procedures in performing tasks. In this study, the researcher had an opportunity to analyse the SOP in continuous development, testing and deployment of software. The organisational structure was also closely analysed to understand the various silos and how they interacted with each other.

3.5.2 Structured Interviews

According to Kothari (2004), structured interviews as a set of questions deliberately crafted with a common goal or objective to collect information through oral or verbal communication. An interview provides a two-way verbal communication direct integration between the researcher and their research respondents or participants, and can be face to face, over a phone call or online meeting. The researcher used structured

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

interviews in the extraction of primary data from their research respondents and this enabled them to stimulate responses from the research participants who were the staff deployed to the development, testing, deployment at Ecocash. The interviews were semi-structured and the researcher had interview schedules to help in remembering and ordering the questions. According to Leedy and Ormrod (2005) semi-structured interview are based on the use of an interview guide; a list of questions or topics to be covered by the interview. The researcher made appointments with the selected key informants and made arrangements on the dates and times and mode of the interview. The interview sessions were an average of 40 to 60 minutes each. The advantages of semi-structured interviews is that they give the researcher an opportunity to clarify some questions for the interviewee, make follow up questions to probe and expand the interviewees' responses.

Interviews with four experts who work with the current system were undertaken to gather information on the processes, procedures, experiences, and issues of the current Ecocash manufacturing line. Four significant individuals who were specifically chosen for their knowledge, organisational roles, and professional interest in this research participated in the interview process. The Secure Development Coordination unit conducted the initial interview with a senior expert whose insight into the organization's current frameworks and criticism of the newly proposed framework were used to enhance each iteration of the framework. The purpose of the interview was to learn what stakeholders expected from a new framework and how they thought the current environment would be enhanced.

A key member of the operations team who participated in the second interview was also considered to be a key informant because of their expertise in operations and knowledge of how the business applied the Continuous Integration and Continuous Deployment model in its actual day-to-day working environment. The primary goals of the interview were to comprehend how and when CI/CD was used, become familiar with the Ecocash CI/CD tools, comprehend the security measures currently in place for CI/CD environments, and possibly identify as well as recommend changes that could be very significant in narrowing the scope for the framework to be designed.

To gain an understanding of some security baselines regarding CI/CD, DevOps streets, and development at some point in the past based on their experiences at Ecocash of overall experience in the past, the third interview was held with a member of the management of Security and Privacy. This individual is responsible for ensuring that the teams working on projects use the frameworks. The major goal of the interview was to learn about their daily routines, roles, and the deployment and testing frameworks they currently employ.

A representative from the quality and compliance unit, which is in charge of ensuring that security requirements and best practises are followed, participated in the fourth interview. The purpose of this interview was to learn more about the existing state of the security of CI/CD pipelines, how the new framework would

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

benefit them, and to find CI/CD experts with whom the researcher could speak and potentially get some insightful interactions. No additional interviews were conducted until after the first version of the framework had been delivered since the researcher considered that the information gathered in the four interviews was sufficient for the study's goals.

The researcher also had the chance to examine a real-world application of the company's CI/CD process, which she used to construct a sample project and learn how each of the solution's components functions separately and in concert. The example project involved adding some code to the version control system, automating the building of new code, and evaluating the code's quality using a static code analyser.

3.5.3 Experiments

The researcher used the experiential methods to make decisions on how to make improvements on existing systems utilising a process that relies on personal judgement of the professional practitioner utilising their personal experience or utilising a trial and error method. In this instance, the methodology also included the use of formally observed quantitative data as well as additional practical observations that were fused into the overall perception based on the extracted attitudes and beliefs that was predominately composed of Ecocash's qualitative experience outputs. Al-Ghamdi (2019) observed that many consultants and the majority of managers use this mind-set while making decisions. Implementing this strategy may result in "satisficing" solutions, meaning solutions that are adequate but not perfect, which combines the concepts of satisfying and optimising.

The process of crafting this framework included a series of trials of proposed system designs leading to the perfection of the final framework. To determine how much the artifact solves the issue, experiments were conducted. In order to develop the framework and make it better than the current systems, the interviewers and other experts provided advice and critiques to the trial designs. The process included repeating the "design, development, and assessment" action when the experts would have offered more advice and input for enhancements to the artifact, until the researcher feel the system is now perfect for the current environment as it continuously change and more changes would be needed in the future. Additionally, a user evaluation was carried out at Ecocash used as the case study. This evaluation's objective was to remodel the artifact in light of its usage environment.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

The following deployment automation was adopted from the Development side in the experimentation.

1. Git Branching System

There is need to have a branching system on git where there must be the Staging branch and the Production branch

- For purposes of security the Development manager should be the only one allowed to approve merge requests into the Staging branch
- Then in the Production branch the Service delivery manage has to approve merge requests after the Staging has been completed and the pipeline goes to Production

2. Configurations

- The proposal is to have a git project with all the application configs (e.g. application. Properties).
- DevTeam to have their applications configured in such a way that they point to the configurations in the git project.

3. Deployment

The application will be deployed on a Kubernetes cluster which makes use of docker for building

Images and Kubernetes for orchestration.

- The DevTeam to agree with DevOps to come up with a docker file that will package the application into an image
- The docker image is then pushed to a registry where it is stored for version controlling purposes
- Upon completion the docker image is then deployed to Kubernetes cluster and a service is configured and exposed
- The CI/CD pipeline process will be handled by Jenkins server

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

Below is the proposed architectural diagram for CD/CD framework, using ECOCASH as the case study.

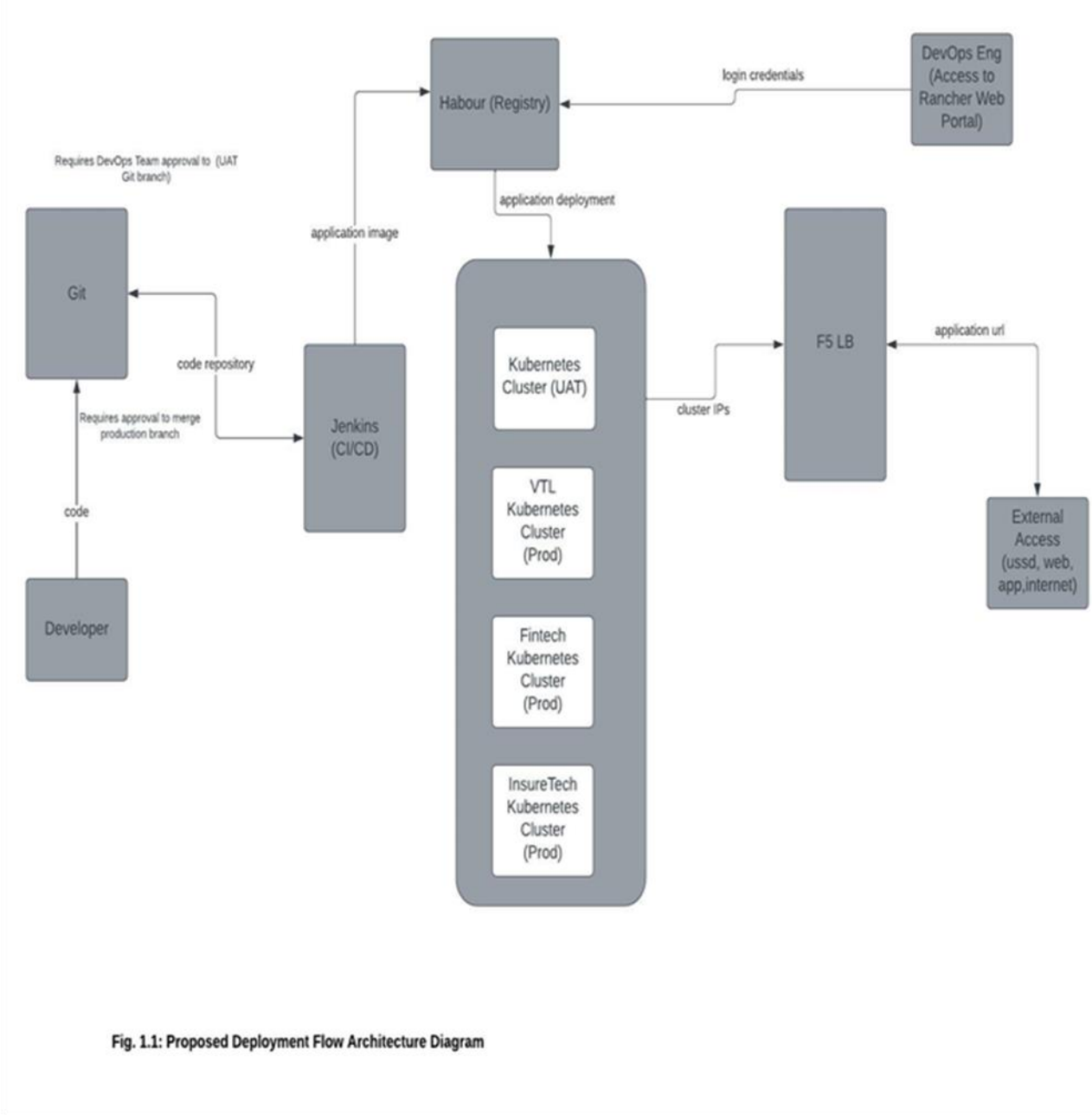


Fig. 1.1: Proposed Deployment Flow Architecture Diagram

Notes

F5 – load balancer

Git – repository for version control

Following serves as basis to adopt this framework:

- Anyone can deploy
- Faster, more efficient deployment

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

- Increased productivity
- Fewer errors
- More frequent releases
- Immediate feedback

3.6 Ethical consideration

By requesting authorization from the relevant authorities to collect data in order to examine the company, the ethical criteria of research were properly observed. The approval letters are appended as Appendices I and II at the end of this report. Before their agreement was requested, the participants were also made aware of the nature and goal of the study and told that the information acquired would only be used for academic purposes. They were also assured of the confidentiality and anonymity of their comments. In their paper, Leedy and Ormrod (2005) divided ethical concerns in research into four categories: informed consent, right to privacy, candour with peers in the field, and protection from damage in any form. By making respondents aware of the goal of the research, the researcher adhered to all professional research practises strictly and consistently and protected the privacy and confidentiality of the respondents as well as protecting the integrity of the data collected. The privacy of the company was respected and the information gathered managed in a manner that would not compromise the privacy or release such information to competitors.

3.7 Chapter Summary

The chapter presented and explained the methodology or process followed in crafting the framework. The design science research framework was utilised to guide the process which analysed the case study of Ecocash Holdings. The problem was analysed by document analysis as well as interviews with practitioners

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

CHAPTER 4

METHODS/FINDINGS

4.0 Introduction

This section makes a presentation of the resultant framework and solution to the problem the project sought to solve.

4.1 Methods/findings:

Based on experimental results, we demonstrate the advantages of for Continuous Integration, Deployment and Testing. We demonstrate how the organisation can make use of the DevOps framework to speed up software deployments. To do some POCs and demonstrate the use of this framework, it is not practical to setup a full Kubernetes cluster with setup of nodes and clusters, we will use MINIKUBE which has option to have Kubernetes locally. Minikube is a single node cluster and the easiest way of deploying Kubernetes for learning, demonstrating and POC purposes. In Minikube the master and worker processes are on the same machine and supports most of the Kubernetes feature

This illustration is working on the assumption that, Minikube is installed, Jenkins, docker.

I will now deploy a spring boot application with 3 replicas to minikube. The spring boot application is a simple Rest controller which returns a “Hello Everyone message”. The process / steps we will follow are depicted below

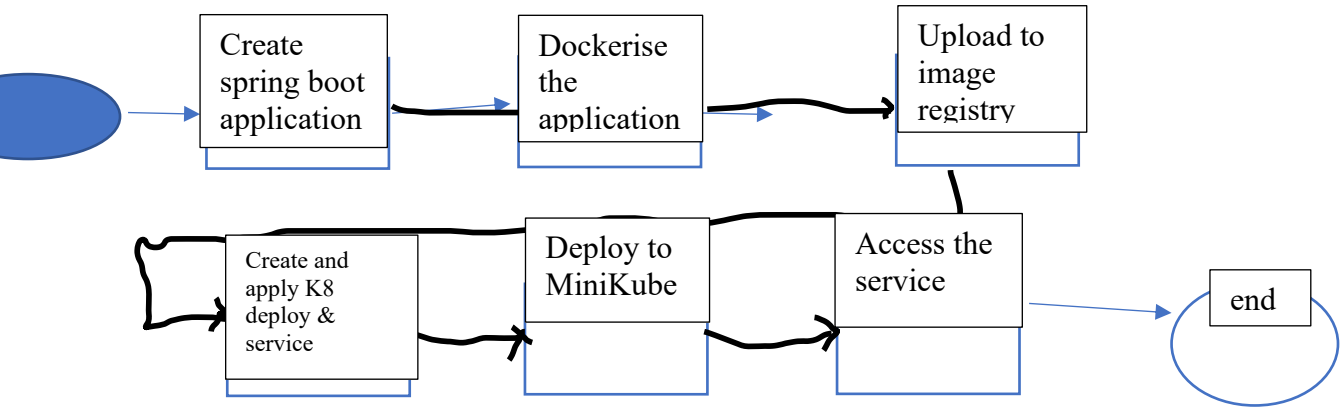


Fig: 4.1 Application Process flow

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

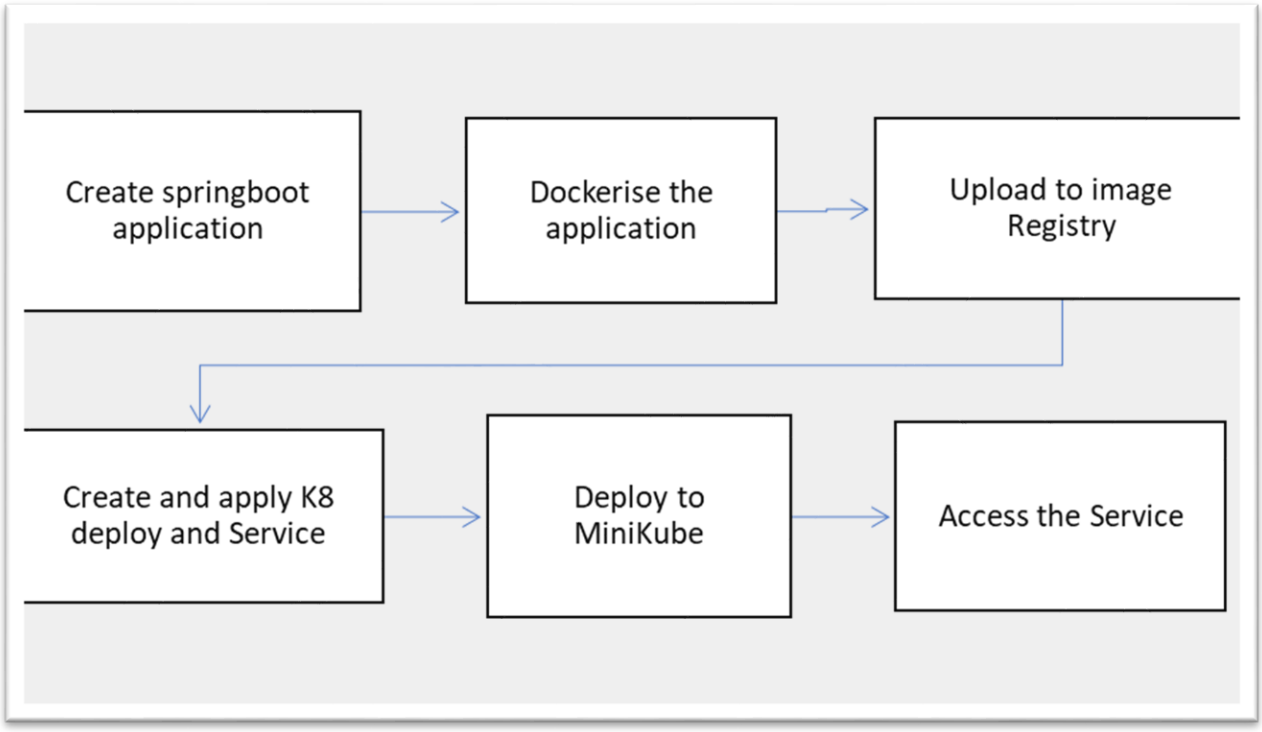


Fig 4.2 Application deployment process flow

4.2 Spring boot code

First step is create a spring boot application service

```
package com.raj.nola.hello;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class HelloWorldSvcApplicationTests {

@Test
void contextLoads() {
}

}
```

Docker file

Dockerise the application using the below file and upload to the repository

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

```
FROM java:8-jdk-alpine
COPY ./target/hello-0.0.1-SNAPSHOT.jar app.jar
CMD [ "sh", "-c", "java -jar /app.jar" ]
```

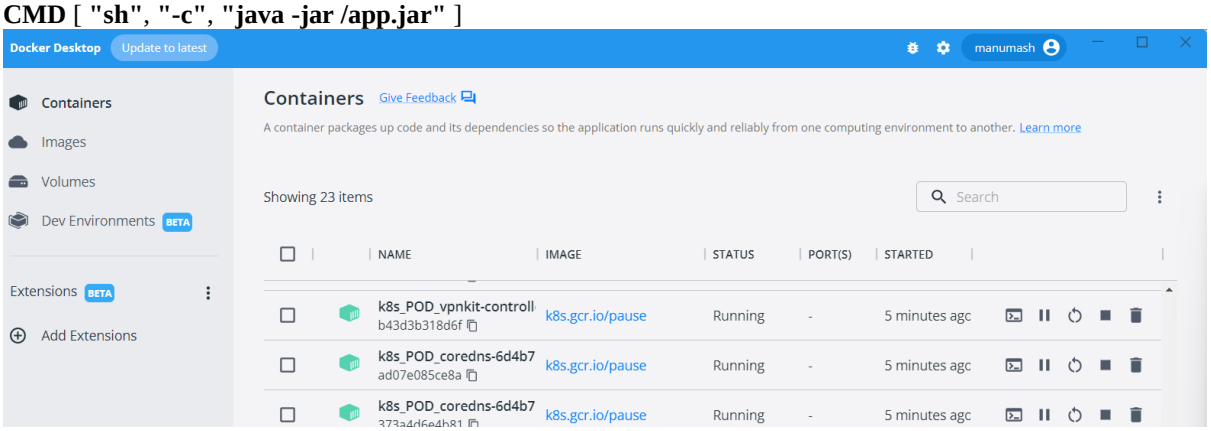


Fig 4.3 docker container dashboard

Deployment File

```
apiVersion: apps/v1
kind: Deployment # Kubernetes resource kind we are creating
metadata:
  name: hello-world-svc
spec:
  replicas: 3  #Number of replicas that will be created for this deployment
  selector:
    matchLabels:
      app: hello-world-svc
  template:
    metadata:
      labels:
        app: hello-world-svc
    spec:
      containers:
        - name: hello-world
          image: raje/hello-world # Image that will be used to containers in the cluster
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 8080 # The port that the container is running on in the cluster
```

Service to expose

```
apiVersion: v1 # Kubernetes API version
kind: Service # Kubernetes resource kind we are creating
metadata:
  name: hello-world-svc
  labels:
    name: hello-world-svc
spec:
  ports:
    - port: 8080 # The port that the service is running on in the cluster
      targetPort: 8080 # The port exposed by the service
      protocol: TCP
      nodePort: 31000
  selector:
    app: hello-world-svc
  type: NodePort # type of the service. LoadBalancer indicates that our service will be external.
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

Start Minikube and deploy the application either from the dashboard or through the command line

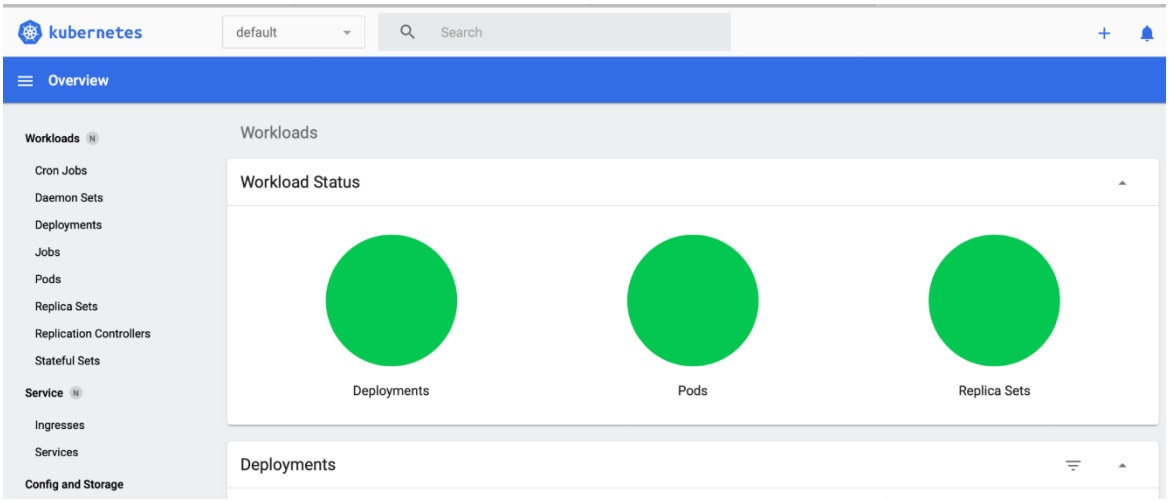
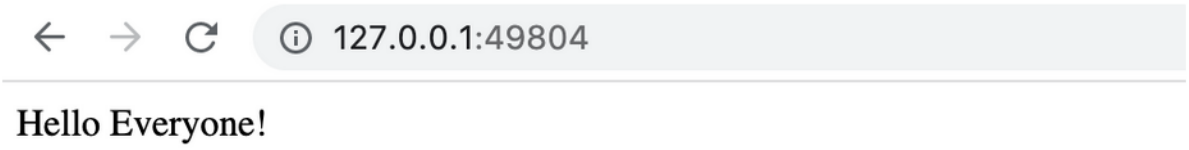


Figure 4.4 Kubernetes dashboard

The test the deployed application go to browser as below



4.3 Economic analysis

The framework designed does not require an investment into new hardware such as servers and network accessories as they are already available. May also make use of the current human capital resources in terms of the developers, testers and operations teams. The system could also be deployed on existing cloud infrastructure accessible by all other teams at minimal costs.

4.4 Environmental assessment

The framework for automation and automated designs has no direct negative environmental effects

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

CHAPTER 5

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

5.0 Introduction

This section now concludes and summarises the whole process leading to the final design of the recommended framework.

5.1 Summary

In the recent past, software release has been going through some major transformation. With software development becoming everything, most software developers are now trying to adapt to this high demand, and automation has become inevitable. Most organisations have adopted the use of pipelines of continuous Integration (CI) and continuous development (CD). This has been developed because of the over demanding for deployment and delivery of new software releases. This faster and continuous software delivery is made possible through DevOps. It uses the Agile methodology in which developers work together closely with infrastructure engineers to achieve the faster delivery, and deployment of applications more quickly and reliably. This Organisational Framework for Continuous Integration, Deployment and Testing in DevOps Environment overcomes the delivery challenges by improving the time to market, testing of release, availability. Primary data was gathered from interviewing practitioners to get a closer look at the operational challenges and their expectations from a new framework. Based on experimental results, we demonstrate the advantages of for Continuous Integration, Deployment and Testing. This solution provides an effective and efficient way to generate, manage, customize, and automate Agile-based CI and CD framework through automated pipelines. This is demonstrate using a Kubernetes cluster using docker containers. The cluster works as a container orchestration tool. Rancher cluster manager is used to manage the cluster. Changing the principles of this solution and expanding it into multiple platforms (windows) will be addressed in a future discussion.

5.2 Conclusions

Following serves as basis to adopt this framework:

- Anyone can deploy
- Faster, more efficient deployment
- Increased productivity
- Fewer errors
- More frequent releases
- Immediate feedback

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

5.3 Recommendations

It can be recommended that the DevOps team adopt the system and fuse it with the organisational structure and ensure there is teamwork and effective flow of information and skills between the development, testing and operations teams

5.4 Contributions

The framework was anticipated to contribute to the improvement of the DevOps process as well as the CD and CI processes

5.5 Limitations

The development of this framework was limited to a single organisation with different needs and structure and therefre, the resultant framework maybe difficult to deploy in other different settings.

5.6 Limitations of proposed solution

This method may not be easy to apply to complex systems that require the discipline and collaboration that DevOps provides. Such systems typically have multiple software and hardware components, which may have different delivery cycles and probably timelines. This framework may not fully facilitate the coordination of some delivery cycles and system-level release planning.

It is difficult to implement full continuous deployment due to the software's crucial use case. Customers are hesitant to upgrade their software when there is a chance that it will break following the update. It takes more work to apply bug fixes and manage the CI pipeline for each version of the product because many versions are being maintained at once. Additionally, when customer-specific modifications have been made to the software, automating upgrades between versions is challenging. The difficulty of updates may be made easier by introducing a more rapid release cycle, which would lessen the number of changes between software versions.

5.7 Limitations of the whole research

When changes happen quicker than lessons can be learned, the capacity to use trial and error learning to fine-tune the performance of information systems essentially becomes meaningless. A rigorous prediction methodology based on cause and effect understanding is now more important than ever. One is entitled to conclude that the decision-making process can be enhanced by substituting a more formal, experimental, quantitative, scientific, and "optimising" approach for the experiential, qualitative, judgmental, and satisficing approach.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

5.8 Further research

The framework was designed based on data gathered from a single entity, and therefore may have structural issues in deployment in other institutions. Therefore, a framework may also be crafted based on data gathered from multiple institutions to ensure some structural and operational differences are also incorporated in the design.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

REFERENCES

[1] Al-Ghamdi, K. A. (2019) 'Improving the Practice of Experimental Design in Manufacturing Engineering', (May).

[2] Amaradri, A. S. and Nutalapati, S. B. (2016) 'Continuous Integration, Deployment and Testing in DevOps Environment', *Thesis*, p. 115.

[3] Bheri, S., Vummenthala, S. and Molleri, J. S. (2019) 'An Introduction to the DevOps Lifecycle', (June).

[4] Blomberg, V. (2015) 'Turun kauppakorkeakoulu Turku School of Economics'.

[5] Bobrovskis, S. and Jurenoks, A. (2018) 'A survey of continuous integration, continuous delivery and continuous deployment', *CEUR Workshop Proceedings*, 2218, pp. 314–322.

[6] Bowen, G. A. (2011) 'Document Analysis as a Qualitative Research Method', *Qualitative Research Journal,* 9(2).

[7] Chandramouli, R. (2022) *Implementation of DevSecOps for a Microservices-based Application with Service Mesh*, *NIST Special Publication*.

[8] Creswell, J. W. and Creswell, J. D. (2018) *Research design : qualitative, quantitative, and mixed methods approaches*. SAGE.

[9] Donca, I. C. *et al.* (2022) 'Method for Continuous Integration and Deployment Using a Pipeline Generator for Agile Software Projects', *Sensors*, 22(12). doi: 10.3390/s22124637.

[10] Ebert, C. *et al.* (2016) 'DevOps', *IEEE Software*, 33(3), pp. 94–100. doi: 10.1109/MS.2016.68.

[11] Gebril, A. (2020) 'Ahmed Gebril CONTINUOUS INTEGRATION , A LI- TRERATURE REVIEW', (April).

[12] Gokarna, M. and Singh, R. (2021) 'DevOps: A Historical Review and Future Works', *Proceedings - IEEE 2021 International Conference on Computing, Communication, and Intelligent Systems, ICCCIS 2021*, 2001, pp. 366–371. doi: 10.1109/ICCCIS51004.2021.9397235.

[13] Helsinki, A. W. and Wikström, A. (2019) 'Benefits and challenges of Continuous Integration and Delivery-A Case Study'.

[14] Henning, S. and Hasselbring, W. (2021) 'The Titan Control Center for Industrial DevOps analytics research', *Software Impacts*. Elsevier B.V., 7(December 2020), p. 100050. doi: 10.1016/j.simpa.2020.100050.

[15] Hochbergs, E. and Sjödahl, L. N. (2020) 'Software Configuration Management in a DevOps context', *Lu-Cs-Ex*.

[16] Hornbeek, M. (2018) 'DevOps Testing - The Primary Key to DevOps and Continuous Delivery', p. 18.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

[17] Kothari, C. R. (2004) *Research Methodology: Methods and Techniques*. New Age.

[18] Kozludzhova, K. (2020) 'Research Methodology for Studying Innovations in the Software Industry', 9(06).

[19] Leedy, P. D. and Ormrod, J. E. (2005) *Practical research : planning and design*. Pearson.

[20] Luz, W. P., Pinto, G. and Bonifácio, R. (2019) 'Adopting DevOps in the real world: A theory, a model, and a case study', *Journal of Systems and Software*. Elsevier Inc., 157, pp. 1–16. doi: 10.1016/j.jss.2019.07.083.

[21] Menzel, G. and Macaulay, A. (2015) 'DevOps - The Future of Application Lifecycle Automation', *Capgemini.Com*, p. 24.

[22] Rejström, K. (2016) 'Implementing Continuous Integration in a Small Company: A Case Study', p. 107.

[23] Riti, P. (2018) 'Introduction to DevOps', *Pro DevOps with Google Cloud Platform*, pp. 1–18. doi: 10.1007/978-1-4842-3897-4_1.

[24] Scott, E. (2021) 'Adopting DevOps Practices : A Case Study'.

[25] Shahin, M., Ali Babar, M. and Zhu, L. (2017) 'Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices', *IEEE Access*, 5, pp. 3909–3943. doi: 10.1109/ACCESS.2017.2685629.

[26] Sharma, S. and Coyne, B. (2015) *DevOps for dummies*. 2nd IBM Li. New Jersey: Wiley.

[27] Truong, H. L. and Klein, P. (2020) 'DevOps Contract for Assuring Execution of IoT Microservices in the Edge', *Internet of Things (Netherlands)*. Elsevier B.V., 9, p. 100150. doi: 10.1016/j.iot.2019.100150.

[28] Wahaballa, A. *et al.* (2015) 'Toward unified DevOps model', *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, 2015-Novem, pp. 211–214. doi: 10.1109/ICSESS.2015.7339039.

[29] Wiedemann, A. *et al.* (2020) 'Understanding how DevOps aligns development and operations: a tripartite model of intra-IT alignment', *European Journal of Information Systems*. Taylor & Francis, 29(5), pp. 458–473. doi: 10.1080/0960085X.2020.1782277.

[30] Yang, Dawei *et al.* (2020) 'DevOps in practice for education management information system at ECNU', *Procedia Computer Science*. Elsevier B.V., 176, pp. 1382–1391. doi: 10.1016/j.procs.2020.09.148.

[31] Sharma, S.: DevOps for Dummies. John Wiley & Sons, Inc, New Jersey (2015)

[32] Erich, F., Amrit, C., Daneva, M.: Report: DevOps Literature Review. Twente: University of Twente (2014)
Hüttermann, M.: DevOps for Developers. Apress, New York (2012)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

APPENDICES

Appendix I – Permission LetterAppendix II – Structured Interview Guide

(Introduction and consent information)

What are your key responsibilities – Regarding the delivery of software?

What do you see as the benefits of CD?

What are challenging aspects of CD? – Implementation – Organisational

How has the CD process evolved over time?

What challenges have you observed with the current implementation of CD in the company?

How do you think those challenges maybe solved?

Any other comments about manual or automated CD?

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 05, May-2023

Appendix III – Proposed Deployment Flow Architecture Diagram