DEVELOPMENT AND IMPLEMENTATION OF AN AUTOMATED BUILD PIPELINE MODEL FOR PRIVATE NETWORKS

SAAIMAH PATEL

ZCAS UNIVERSITY

2023

DEVELOPMENT AND IMPLEMENTATION OF AN AUTOMATED BUILD PIPELINE MODEL FOR PRIVATE NETWORKS

SAAIMAH PATEL

A Final Year Research Project submitted in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science

ZCAS University

2023

DECLARATION

Name: Saaimah Patel

Student Number: G18021

I hereby declare that this final year research project is the result of my own work, except for quotations and summaries which have been duly acknowledged.

Plagiarism check: %

Signature:  Date: 31 December, 2023

Supervisor Name: Aaron Zimba

Supervisor Signature:

Date:

ACKNOWLEDGEMENT

I am grateful to express my deepest appreciation to my supervisor, Dr. Aaron Zimba, for his invaluable guidance, patience, and advice throughout this project. His expertise, support, and constructive feedback have been instrumental in its successful completion. I am thankful for his dedication, availability, and mentorship, which have greatly contributed to my academic growth. I am truly grateful for the opportunity to have worked under his supervision.

THANK YOU.

DEDICATION

I would like to take this opportunity to express my heartfelt gratitude and appreciation to my family. Their unwavering support, encouragement, and love have been the foundation of my success throughout this journey. Their understanding, patience, and belief in me have provided the motivation and strength to overcome challenges and pursue my goals. I am truly grateful for their constant presence, guidance, and unwavering belief in my abilities. Their support has been invaluable, and I am forever grateful for their unwavering support and love.

ABSTRACT

This research investigates the creation and application of Automated Build Pipelines (ABP) in private network settings with the goal of developing a model or framework that would manage semantic versioning, increase operational efficiency, and reduce downtime. In addition to identifying and analysing the risks and challenges, the research offers insights into potential roadblocks and weaknesses. A thorough framework or model is created by the discovery of knowledge gaps and restrictions to direct the evaluation, planning, and implementation of the automated build process. A conceptual framework establishes the principal characteristics of the proposed ABP.

Google Sheets and Google Looker Studio are used to show how the original concept is practically implemented, taking into account the risks and obstacles that have been identified. The performance and efficacy of the used model are evaluated after a careful analysis and assessment of the data. By offering useful advice and lessons learned to companies looking to deploy automated build pipelines on private networks, the study adds to the body of knowledge. All things considered, this work contributes to the field's understanding and offers insightful advice for implementing automated build pipelines in private network environments successfully.

Key words: Automated Build Pipelines, Semantic Versioning, Continuous Integration (CI), Continuous Deployment (CD)

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Automated Build Pipeline – ABP

Continuous Integration - CI

Continuous Deployment – CD

Open Source Software - OSS

Software Development Life Cycle - SDLC

Extreme Programming - XP

Lean Software Development - LSD

Feature-Driven Development - FDD

CHAPTER 1 - INTRODUCTION

1.1 Background to the study

The need for safe and effective private networks has increased at an unprecedented rate due to the constantly changing field of information technology. Organisations are always looking for novel ways to simplify network infrastructure administration since they are compelled to safeguard critical data. Investigating Automated Build Pipelines (ABPs) as a workable strategy for enhancing efficiency in private networks is one interesting direction.

The goal of this study is to thoroughly examine whether or not ABP implementation is feasible in the setting of private networks. Because ABPs automate many crucial components of network management, including as configuration, security enforcement, and software updates, they offer a revolutionary paradigm shift. ABPs seek to minimise human error, lower operating costs, and improve network stability by substituting automated procedures for manual interventions (Smith et al., 2017).

This research recognises the difficulties in protecting private networks, which call for careful preparation, conformity to regulations, and the implementation of strong security measures . Organisations may be able to develop a more thorough and secure network architecture by including these factors into the ABP framework.

The strong need for automation approaches derives from the rapid and effective development of software, particularly for cloud-based systems. The advent of Continuous Integration/Continuous Delivery (CI/CD) offers a series of procedures for the automated development, testing, and rollout of new software. Consequently, in order to automate the development and deployment of new software and apps, many businesses integrate CI/CD pipelines into their platform (Bello *et al.*, 2022).

1.2 Problem Statement

Private networks are becoming increasingly in demand at a time of constant technological advancement. Enterprises, especially those managing confidential data, look for reliable and effective ways to protect their information (Steffens, Lichter and Döring, 2018).

Simultaneously, network infrastructure management has become increasingly dependent on the requirement for automated and efficient processes (Debroy and Miller, 2020). The primary goal of this study is to determine whether it is feasible to use an Automated Build Pipeline (ABP) to improve the performance of private networks.

Since the introduction of cloud service providers' pay-as-you-go models, corporations are no longer needed to make an upfront commitment to purchase pricey hardware and the first step of establishing the infrastructure. The time needed to deploy these computational resources has decreased from days to a few minutes thanks to the usage of virtualization and orchestration. That is insufficient, nevertheless, to develop a cloud-based infrastructure (Garg and Garg, 2019)

The way network infrastructure is managed nowadays entails manual interventions, which might result in security flaws and human mistake. The industry practice for managing code is to use a cloud-based, secure git service (usually Github). Deploying platforms is usually a tedious, time-consuming, manual procedure that involves installing several private and secure physical or cloud servers using virtual private networks (VPNs), setting up the environment, and then releasing the code. Every time there is a modification, the process is repeated. This has a major effect on the software development team's speed and agility. (Mysari and Bejgam, 2020).

(Zampetti *et al.*, 2020) claims that industrial organisations who implemented continuous improvement (CI) reported significant increases in productivity, customer satisfaction, and the ability to provide high-quality products rapidly through iterative procedures. The continuous integration (CI) methodology is one of the most widely used software engineering approaches because of its indisputable advantages, which have also attracted many Open Source Software (OSS) engineers to adopt it. Even while continuous integration (CI) is becoming more and more common, its widespread use makes it challenging to implement in traditional development contexts. As such, in their most recent research, scholars have examined the challenges and issues associated with the transition to CI. They found that two of the difficulties developers have are identifying build issues and automating the build procedure.

1.3 Aim

The aim of this research project is to develop and implement an automated build pipeline for private networks.

1.3 Objectives of the study

To achieve the aim above, these are the objectives set:

1. To identify the challenges and risks associated with implementing automated build pipelines in private networks in order to use them to develop a framework.
2. To develop a framework for assessing, planning, and executing the implementation of an automated build pipeline for private networks with the goal of achieving operational efficiency, handling semantic versioning, and minimising downtime.
3. To develop and implement an automated build pipeline model for private networks
4. To evaluate the developed model

1.4 Scope and Limitation

The frameworks and pipeline model development for an automated build pipeline for private networks will be covered in this project. It will also include the identification of security risks and issues that impact the pipeline. However, there will be no deployment and machine learning involved. It has been deemed to exceed the scope of this project

1.5 Significant of the Project

In an era of increasing cyber threats and data breaches, this study is significant because it explores innovative ways to enhance automated deployment within private networks. The findings could contribute to more reliable procedures and quicker implementation. Additionally, this research could greatly increase an organisation's operational efficiency. Businesses may limit human errors, cut down on downtime, and save time and money by automating deployment operations.

Automation can result in significant cost savings for companies if it is shown to be more economical than manual techniques. This is especially important in situations where funds are limited. Best practices and guidelines for deploying automated build pipelines in private networks can also be developed with the aid of this research. For businesses wishing to implement automation, these guidelines can be a great tool for ensuring a safe and seamless transition (Lunde and Colomo-Palacios, 2020).

By addressing gaps in the existing literature and conducting rigorous research, this study can make significant contributions to the academic field. It can expand the body of knowledge related to private networks, automation, and their intersection, serving as a reference for future researchers.

Private networks are essential to the operations and data security of organisations in a variety of industries. The researcher's conclusions can directly help these businesses by offering guidance on how to improve integration and deployment.

The suggested ABP would automate crucial network deployment procedures like software upgrades, configuration management,continuous integration and deployment in order to lessen these problems. Network reliability will be improved and operational overhead will be decreased by this automation. This project aims to automate this process using the CI/CD technique. After changes to the source code are merged into the main branch, versioning and platform deployment are meant to be handled by a "orchestrator" (Pinto *et al.*, 2018) As mentioned in With continuous integration and delivery (CI/CD), deploying a system on a regular basis necessitates extensive post-deployment attention to the most recent system release or update. One of a product's most important lifecycle components is continuous development, delivery, and deployment. (Dwivedi, Semalty and Moondra, 2021)

Software nowadays is created quickly. Organisations mostly rely on automated build, test, and release procedures to maintain that fast pace. In order to achieve this, developers can make incremental codebase changes, which are then collected, linked, and packaged into software deliverables, tested for functionality, and distributed to end users by Continuous Integration and Continuous Deployment (CI/CD) services (Gallaba, 2019)

Workflows for Continuous Integration, Delivery, and Deployment are dictated by both technological needs and organisational cultures and preferences (Railić and Savić, 2021). This research offers a method for creating and executing a continuous integration/delivery (CI/CD) process that adheres to best practices for the build, test, deploy, and release phases. Additionally, it explains the difficulties in designing CI/CD systems and suggests possible solutions.

1.6 Research Questions

Below are the research questions:

- What are the challenges and risks associated with implementing automated build pipelines in private networks?
- How do these risks and challenges impact the development of the framework?
- What is the significance of this project?
- What metrics will be created and used to evaluate the model created?

1.7 Preliminary sections of the project report

These are the sections below:

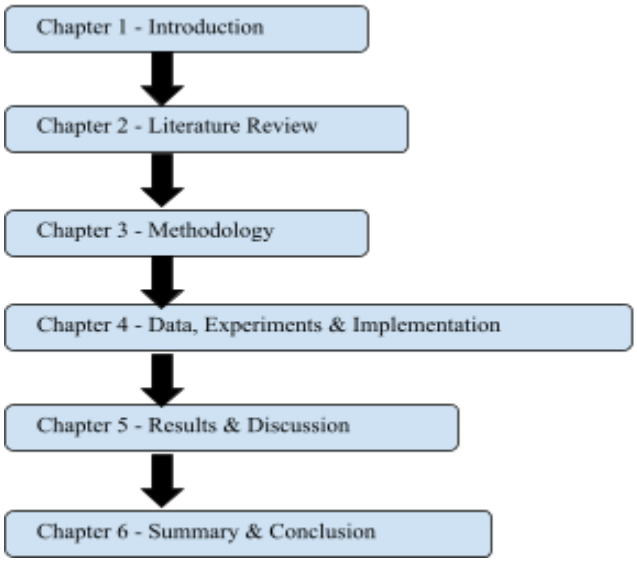

Figure 1.1: Flow Diagram of sections of the report

CHAPTER 2 - LITERATURE REVIEW

2.1 General Background

Examining the current issues in private networks is essential to comprehending the context of automating build pipelines for networks. These difficulties frequently include the complexity of network infrastructures, the necessity for quick responses to accidents and vulnerabilities, and the sophistication of cyber threats (Riti, 2018)

Automation has become more popular as a solution for these problems. The advantages of automating deployment tasks have been emphasised by research in this field (Mitesh S., 2015). Nonetheless, there is a lack of information in this literature about its application to private networks because it mostly concentrates on general networks. Though automation in networking is becoming more and more common, less research has been done especially on private networks. Research on the benefits and difficulties of integrating automation in private network environments are hard to come by.

This research study emphasises the growing importance of automation in networks, particularly in private network environments. Existing research discusses the benefits and challenges of automation, but more specific research is required in the context of private networks.

2.2 Broad literature review of the topic

The methods used in traditional software development are inadequate to meet the demands of modern enterprises. Software development organisations find agile methodologies attractive because they enhance the Software Development Life Cycle (SDLC)'s flexibility, efficiency, and speed. Applying the twelve AgileManifesto principles to methods like Extreme Programming (XP), Scrum, Kanban, Crystal, Lean Software Development (LSD), and Feature-Driven Development (FDD), one can establish the integrity of procedures and practices as well as Agile Project Management. Using an agile methodology with a Continuous Integration (CI) pipeline has improved productivity and sped up software delivery (Arachchi and Perera, 2018).

As mentioned in (Shahin, Ali Babar and Zhu, 2017), Continuous Integration (CI), Continuous Delivery (CDE), and Continuous Deployment (CD) are strategies that assist

organisations in accelerating the development and delivery of software products while maintaining quality. These methods offer advantages such as immediate feedback from the software development process and customers, frequent and dependable releases, increased customer satisfaction and product quality, and strengthened relationships between development and operations teams. Adopting continuous practices, on the other hand, is not a simple undertaking since organisational procedures, practices, and tools may not be prepared to support the complex and hard nature of these practices.

Collaboration between multiple organisations in various locations is necessary for large-scale software development, and this can be facilitated by using a version control system (VCS). Git is a source code management system and free, open-source DVCS. Repo is a repository management tool that maintains a secret repo directory with all project names and paths in an XML file and unifies several Git repositories into a single Git repository. Thanks to the guidance of key concepts like build automation, automated testing, and revision controls, continuous integration (CI) has developed into a best practice for software development. The best approaches for maintaining applications' deployability while adhering to strict quality standards are continuous deployment and delivery. (Hung and Giang, 2019)

According to (Jin and Servant, 2020), empirical research has looked into the practice of Continuous Integration (CI) and its costs, with an emphasis on the influence on software quality, productivity, bug-fixing, and testing. The high cost of running builds is a big concern in CI, with large businesses like Google and Microsoft incurring costs in the millions of dollars. Understanding what causes long build durations and utilising machine learning classifiers to forecast test case failures are two ways to reduce the cost of CI. This study, on the other hand, approaches the expense of CI in a different way by lowering the overall number of builds that are done. It also forecasts build outcomes for any type of modification, complementing existing cost-cutting strategies in CI.

(Jin and Servant, 2020) further helps explain compilation, unit testing, static analysis, server problems, architectural dependencies, stakeholder role, programming language, build environment changes, and persistent build breaks are all characteristics of failing builds. Some studies discovered change characteristics that correlate with failure builds, such as the number of commits, code churn, the number of altered files, the build tool, and

statistics on the most recent build and the committer's history.Predicting failed builds has been difficult in industrial settings where continuous integration has not yet been implemented. Cascade classifiers, semi-supervised algorithms, and predictors rely on the outcome of the previous build, although their predictive power may be reduced in cost-cutting situations. SmartBuildSkip, on the other hand, does not predict based on the last build's state and type.

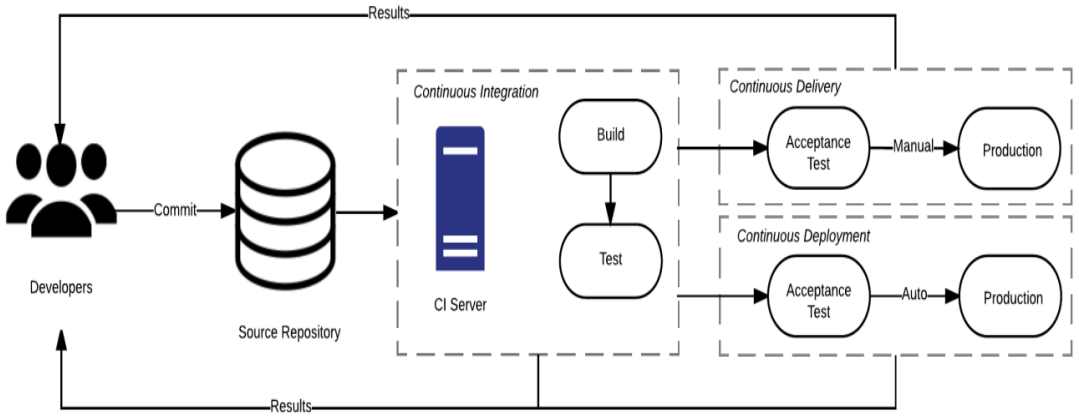

Figure 2.1: The basic operation of CI/CD

2.3 Critical review of related works

One of the works that was found to be similar was “Automated Cloud Infrastructure, Continuous Integration and Continuous Delivery using Docker with Robust Container Security“ by Somya and Satvig Garg. (Garg and Garg, 2019) offers a thorough introduction to container security, CI/CD with Docker, and cloud automation. On the other hand, they did not highlight any similar works done or mention them. It also fails to include a thorough summary of the relevant literature. However, it does acknowledge the important contributions of earlier works. Furthermore, the work fails to point out any shortcomings or gaps in existing literature and fails to explain how it fills these holes.

The part of the paper that discusses using Docker containers to establish continuous integration and continuous delivery, or CI/CD, is quite helpful. The part on Docker Security is a little brief. It would be helpful to have more information on the precise security measures that can be used to protect Docker containers.

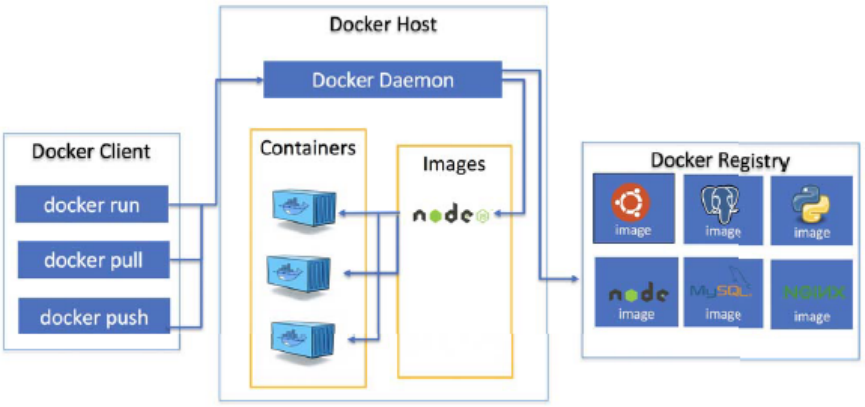

Figure 2.2: The Docker Architecture

Another article that was viewed was the "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management" by Arachchi and Indika Perera. This article focused on expanding the CI/CD pipeline with three automated phases—benchmarking, load testing, and scaling. While the load testing step uses production traffic to obtain accurate performance statistics, the benchmarking phase makes use of a test bench to reduce system downtime. After load testing and benchmarking are finished, the scaling phase is evaluated.

(Arachchi and Perera, 2018) gives a thorough overview of the methodology, including the instruments used, the phases that are sequential, and the rationale for the suggested approach in an effective manner. It also contains relevant performance measurements and a test bench configuration using local virtual instances; nonetheless, the authors note that the simplicity of the application under test has limits. The paper addresses limitations and outlines possible future study while succinctly summarising the main findings and highlighting the benefits of the suggested strategy. Suggestions for enhancements comprise extending the assessment to more intricate apps and cloud-based settings, as well as offering a comprehensive dialogue of obstacles and constraints.

2.4 Comparison with related works

This study was comparable to several publications that had similar research. The table below shows a comparison held between similar research or projects and the proposed framework.

Table 2.1: Comparison with related works

Model	Cost Effectiveness	Merging Conflicts Handling	Semantic Versioning	Failed Builds Handling	Automated Build from git provider	Private Network Applicability
(Pinto et al., 2018)	✓	✗	✗	✓	✗	✗
(Hung and Giang, 2019)	✓	✗	✗	✓	✗	✗
(Garg and Garg, 2019)	✗	✓	✓	✗	✓	✗
(Arachchi and Perera, 2018)	✗	✗	✗	✗	✗	✗
Proposed Model	✓	✓	✓	✓	✓	✓

Choosing a Continuous Integration and Delivery (CI/CD) approach usually involves an issue of cost-effectiveness. Pinto et al., 2018 and Hung an Giang, 2019 excel in this area by highlighting the optimisation of resources and the reduction of infrastructure costs. Arachchi and Perera, 2018 and Garg and Garg, 2019 might not be as cost-effective, though. Garg and Garg's dependence on outside Git providers might raise expenses; nevertheless, Arachchi and Perera, 2018 are unclear in this regard.

Two essential components of CI/CD are versioning and conflict resolution. Here, Garg & Garg, 2019 takes the lead by providing integrated tools to manage codebase conflicts and guaranteeing clarity through semantic versioning support. In order to resolve conflicts, Pinto et al., 2018 and Hung and Giang, 2019 may need to use additional tools or manual intervention, and they do not discuss versioning procedures.

Another fascinating contrast is between network support and build automation. Garg & Garg, 2019 is noteworthy once more since it allows automated builds to be started automatically from Git providers, much like a flowing procedure. The other models could provide potential bottlenecks in the workflow by requiring manual build initiation or not

integrating with certain Git platforms. Furthermore, none of the models specifically address support for private networks, restricting their application to public cloud installations for the time being, which may not be appropriate for enterprises with more stringent security requirements.

Different models handle build failures in different ways. Hung and Giang, 2019 and Pinto et al., 2018 appear ready for obstacles, maybe providing alert systems and automatic remedial measures. There is ambiguity on how Garg and Garg, 2019 and Arachchi and Perera, 2018, handle unsuccessful builds.

The proposed framework provides an affordable means of handling disputes and guaranteeing effective use of available resources. It follows the accepted practice of semantic versioning, which enables several developers to collaborate on the same codebase. The model provides safe access to the software development environment, manages unsuccessful builds, and automates Git builds. Moreover, it facilitates automated builds, which cut down on time spent on manual build procedures and human mistakes.

2.5 Conceptual framework/Theoretical framework

The basis of any Automated Build Pipeline for private networks is a strong conceptual architecture that includes multiple essential components. Together, these components guarantee the scalable, secure, and effective implementation of infrastructure.

Continuously Integrated and Deployed: The Continuous Integration and Continuous Deployment (CI/CD) technique is the foundation of the framework. CI/CD promotes agility and lowers mistake risk by automating the integration of code changes, testing, and smooth deployment. This procedure powers the automation of crucial build pipeline steps and is the basis of the ABP.

Automating the Build Pipeline: The code compilation, testing, and deployment processes are all automated by the build automation process, which is the core of the system. It takes the place of manual interventions. By doing this, human error is eliminated and the time and resources needed for network deployments are greatly decreased. The ABP streamlines the process by integrating software updates, security policy enforcement, and configuration management.

Security: The architecture is integrated with strong network security standards. These procedures protect sensitive data and uphold compliance requirements by ensuring that all automated processes comply with industry-specific norms and regulations. Maintaining a safe and reliable network environment requires including security measures into the ABP from the beginning.

Increasing Operational Effectiveness: Through the automation of repetitive and error-prone tasks, the ABP seeks to dramatically improve operational effectiveness. A well-designed ABP has minimised errors, minimised downtime, and optimise resource utilisation. For businesses in charge of private network management, this means increased efficiency and lower costs.

Scalability for Growth: Scalability has to be considered in the design of the ABP architecture. The automated processes should be flexible enough to accommodate changes in network requirements and complexity without sacrificing performance. Assessing the ABP's scalability guarantees both its long-term sustainability and its ability to adjust to changing needs.

Industry Regulations and Compliance: It is critical for private networks to adhere to industry-specific regulations. In order to guarantee that automated processes comply with legal and regulatory requirements, the ABP framework must be created in line with these regulations. In addition to reducing compliance risks and encouraging responsible network administration, this builds confidence and accountability.

Economic Considerations: Because implementing an ABP requires investment, it is important to determine if it is economically feasible. Organisations thinking about automation can gain important insights by analysing possible cost savings and contrasting them with the cost of using old manual techniques. It is important for the ABP framework to be built with a return on investment in mind, since this will support its implementation and guarantee its long-term viability.

Mixed-Methods Approach: The ABP framework uses a mixed-methods approach to obtain a thorough grasp of the research subject. By fusing qualitative evaluations with quantitative data analysis, this method offers a comprehensive grasp of the security and operational efficiency consequences of automated network deployment. By using this

method, the study explores the ABP's efficacy and effects on private networks in greater detail.

The cornerstone for the investigation and creation of an ABP for private networks is laid by this conceptual framework. The research seeks to improve the management and operation of private networks by tackling each of these crucial components in order to develop a safe, effective, and scalable automated network deployment solution.

2.6 Proposed model/system

Based on the conceptual framework described above, the following methodology for building and implementing an Automated Build Pipeline (ABP) for secured private networks is proposed:

1. Architecture of the System: The ABP will be modular in design, with discrete components for code compilation, testing, deployment, and configuration management. This modularity facilitates customisation and scaling. Throughout the pipeline, a dedicated security layer will be included, enforcing network security procedures and compliance regulations at each level. This layer could include systems for vulnerability detection, access control, and encryption. A central orchestrator will oversee the execution of ABP stages, assuring smooth execution, fault management, and resource optimisation.

2. Integration of Continuous Integration and Continuous Deployment (CI/CD): Code changes will be maintained and versioned in a centralised repository, with any alteration activating automated build triggers. Additionally, a full suite of automated tests spanning functional and security elements will be added into the pipeline. The results of the tests will determine the success of the build and, if necessary, will cause rollbacks. After successful testing, the ABP will deploy the tested build to the target network environment automatically. Configuration management, software updates, and policy enforcement are all part of this.

3. Network Security and Regulatory Compliance: Before deployment, the ABP will use dynamic security analysis tools to uncover vulnerabilities in the build. This analysis will guide security patching and configuration changes .Furthermore, the ABP will use automated audits and reporting mechanisms to continuously monitor compliance with

industry-specific regulations. This ensures that legal and security standards are followed at all times. To protect sensitive data and prevent unauthorised access, communication between ABP components and network devices will use secure protocols such as HTTPS and SSH.

4. Scalability and operational efficiency: The ABP will create thorough data on build progress, resource utilisation, and future concerns. Real-time notifications will notify network administrators of significant events that need to be addressed. The ABP will also allocate and optimise resources dynamically based on build requirements, minimising resource waste and ensuring efficient operation. The ABP will be designed with horizontal scalability in mind, enabling for easy extension to suit rising network size and complexity without sacrificing performance. Horizontal scalability, also known as scaling out, refers to a system's ability to handle increased workload by adding additional machines (nodes) to its existing infrastructure. Vertical scaling, on the other hand, entails adding more resources (e.g., CPU, RAM) to existing computers.

5. Economic Feasibility Study: A complete cost-benefit analysis should be performed to compare the initial investment and continuing maintenance costs of ABP with possible reductions in operational expenses and resource utilisation. To reduce upfront expenses and allow for gradual adoption and adaptation of the ABP within the network environment, consider a phased deployment plan.

6. Mixed Methods Approach to Research: The ABP will be outfitted with sensors that will collect information such as build times, resource utilisation, error rates, and compliance measures. This data will be analysed to determine ABP's operating efficiency and effectiveness. To demonstrate the benefits of ABP implementation, the collected data will be compared to existing benchmarks and performance indicators of typical manual deployment techniques.

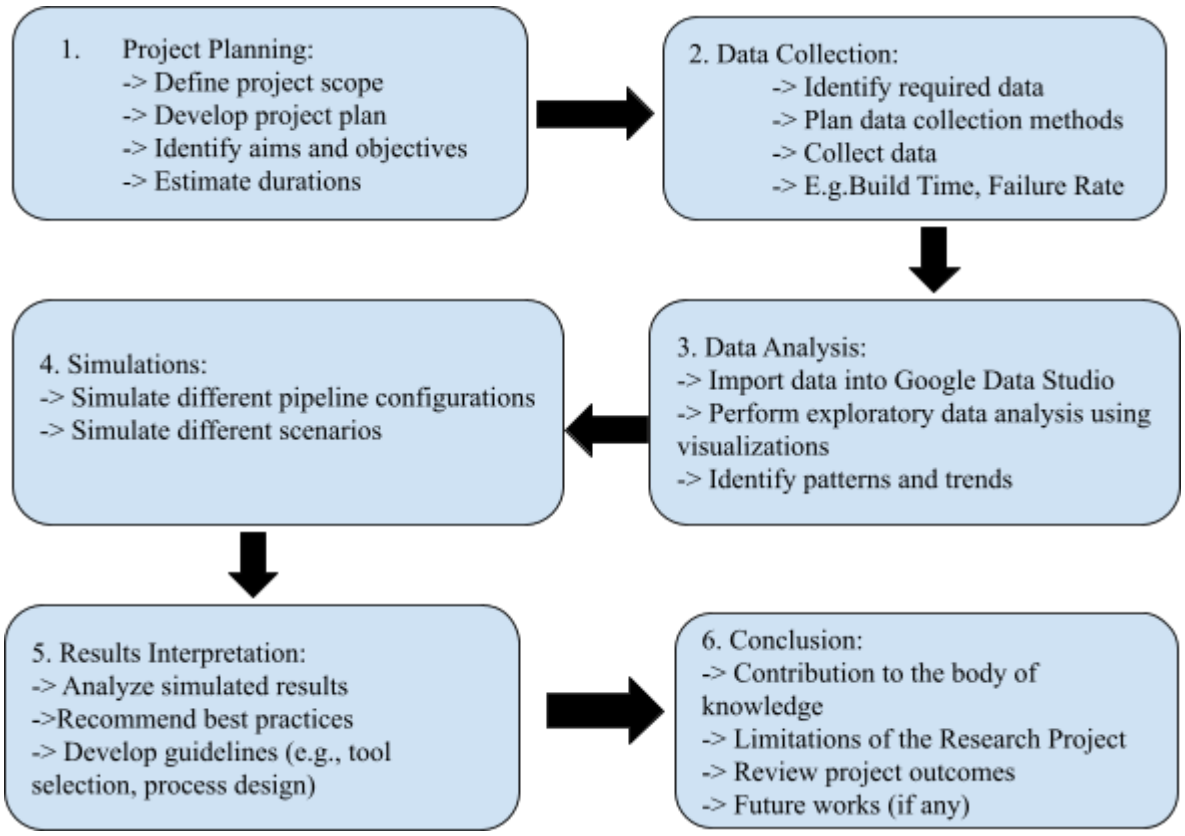

Figure 2.3: Proposed Model

This proposed model offers an overview of the ABP development and implementation process at a high level. It prioritises security, efficiency, and scalability as well as a mixed-methods research methodology for complete evaluation. More research and development is required to fine-tune the individual technological components and implementation methodologies for various private network settings and industry requirements.

2.7 Chapter Summary

This chapter provided a basic background, reviewed the literature, critically analysed related works, and made comparisons between them in order to provide an in-depth basis for understanding the research challenge. Through the identification of knowledge gaps and constraints, this review prepared the way for the creation of a unique ABP framework designed for private networks. The next section of the chapter provided a conceptual framework that outlined the main features of the suggested ABP, with a focus on scalability, automated build pipelines, strong security procedures, and continuous

integration and deployment. The proposed ABP model/system was finally introduced in further depth as the chapter came to a close, laying the groundwork for future development and assessment of it in later chapters

CHAPTER 3 - METHODOLOGY

3.1 Research design

This study looks at how ABPs testing affects important construction metrics using simulated data and a descriptive research methodology. Our goal is to obtain important knowledge about how ABP interacts with different build complexities so that we can improve our build performance tactics.

The main objective is to examine the complex link that exists between ABP setups and critical build parameters, such as resource utilisation, build time, and error rate. Every one of these indications presents a different angle on the stability and effectiveness of the build. This can improve our understanding by analysing data across a range of construction difficulties because different build types may react differently to ABP adjustments.(Lunde and Colomo-Palacios, 2020)

Firstly, we plan to routinely develop simulated data that closely reflects real-world circumstances in terms of precision, ranges, and distributions in order to foster an environment that is favourable for research and experimentation. There are various benefits to this strategy. By overcoming the obstacles to real-world data access, it permits rapid iteration and resource- and logistically-free testing of different ABP configurations. Furthermore, controlled variable manipulation is made easier by simulated data, which allows us to isolate and pinpoint the precise impacts of ABP on build metrics while accounting for outside influences.

It does, however, recognise the inherent drawbacks of simulated data. Even while it helps research and offers insightful information, it is unable to accurately capture the nuanced details of actual construction sites. As a result, we intend to remain open and honest about the simulated nature of the data throughout our investigation and to interpret our results in light of these constraints.

To put it simply, this research methodology uses simulated data to reveal the nuanced effects of ABP testing on building performance. Through an examination of the relationship among ABP, build metrics, and build complexities, our goal is to offer useful

advice for streamlining build procedures and producing software that is dependable and efficient.

3.2 Adopted method and justification

The chosen methodology for this research design is a descriptive one that makes use of simulated data to examine how ABP testing affects important metrics. This section outlines the benefits and drawbacks of the selected approach and justifies its selection.

For this study, a descriptive research approach makes sense since it enables a detailed analysis of the connection between ABP setups and significant build metrics. These metrics, which include build time, mistake rate, and resource utilisation, may be described and analysed to provide a thorough understanding of build stability and efficiency

The use of simulated data allows for the creation of a controlled environment that closely mimics actual conditions. There are various justifications for this methodological decision. First of all, it gets around the difficulties or practical limits associated with obtaining real-world data. The research may swiftly iterate and test different configurations without being constrained by external resource limits by producing simulated data (Koopman, 2019)

Additionally, controlled variable modification is possible when employing simulated data. By taking into consideration potential confusing circumstances, this control makes sure that the effects seen on the build metrics can be specifically linked to the adjustments under test. The study can offer more precise insights into the connection between ABP and the metrics.

This approach is justified by several factors:

- By concentrating on a single case study, the impact of the ABP can be isolated and variables can be better controlled, reducing the impact of outside variables.
- By making use of system logs and performance dashboards that are already in place, less additional data gathering work is required, which improves research efficiency and reduces resource usage.

- Although the measurements and analysis methods utilised are specific to one scenario, they may be applicable to other protected private networks as well, offering insightful information that may find wider applicability.

In conclusion, the chosen study design investigates the effect of ABP performance through the use of simulated data and a descriptive methodology. This approach offers useful insights for streamlining build procedures and producing dependable software, while enabling a comprehensive exploration of the connection between ABP, build metrics, and build difficulties

3.3 Association of research method to project

The project's goals were done by the study method that was selected: a descriptive strategy combined with simulated data. Using an analysis of the interactions between ABP and build time, error rates, and resource utilisation, this method delves into the complex relationship between ABP and chosen metrics. This reflects the project's aim to comprehend the aspects of the ABP across various difficulties, each in a different way that will guide the development of build methods that are optimised.

But the study makes use of simulated data to test the impact of ABP. Here's where simulated data comes in handy: it allows quick testing with ABP setups without the logistical limitations of real-world access. By carefully adjusting the variables, it is possible to determine the exact impact of ABP on construction metrics without using actual data, much like when evaluating real-world data on each factor. This method yields priceless information for enhancing ABP and improving build efficiency.

The research approach is essentially a means of conducting controlled experiments and descriptive analyses. This partnership sheds light on the connections between the ABP and its functionality, thereby coordinating the creation of ideal build procedures and producing outstanding software solutions.

3.4 Research data and datasets

In order to shed light on the complex relationship between ABP and important building performance measures, this study uses carefully constructed simulated data. Our

analysis and insights are based on this data, which is tabulated and displayed in Google Looker Studio reports.

Key performance indicators for seven different pipeline topologies are included in the simulated dataset. These metrics capture the duration of each build process (Build Time), resource utilisation (Resource Utilisation %), frequency of errors (Failure Rate %), response time (Latency), cost per build (Cost per Build), identified security vulnerabilities (Security Incidents), and user feedback on the experience (User Feedback).

We can enable visualisation by connecting Looker Studio to the data source. We are able to perform comparative analyses of performance metrics across configurations, find correlations between variables like resource allocation and security outcomes, and create engaging and educational reports for project stakeholders thanks to this translation of raw data into interactive dashboards and reports. Table 3.1 below illustrates the metadata of the datasource that is connected to Google Looker Studio.

Table 3.1: Metadata description

Data	Meaning
Build	Distinct identity for the execution of a certain build procedure
Time (min)	The build's duration, expressed in minutes
Resource Utilisation (%)	Percentage of resources made accessible and used during the build
Failure Rate (%)	Percentage of builds that were unsuccessful in finishing
Latency (sec)	The amount of time it takes a build to begin processing after it is triggered
Cost per Build (USD)	The amount of money spent on each build's execution
Security Incidents (count)	The amount of instances associated with security that happened during the build
User Feedback	User-provided qualitative input on their impressions of the build process
Pipeline Configuration	A brief description of the particular configuration parameters used in the build process

Using simulated data has a number of clear benefits. First of all, it offers a regulated setting where variables may be accurately adjusted and the distinct effects of ABP changes on metrics can be seen. Second, it enables fast experimentation by enabling the testing and iteration of different configurations without the limitations of real-world implementation. Last but not least, uniformity of the data produced under the same circumstances guarantees precise comparisons and reduces superfluous noise.

3.5 Data collection methods and data analysis techniques

The purpose of this study is to provide light on how ABPs affect performance measures through the use of carefully constructed simulated data. In order to overcome the difficulties in obtaining real-world data and guarantee a safe environment, a specialised computer model simulates actual construction situations and produces detailed data for seven different ABP pipeline configurations. The distribution of resources and security measures differs for every configuration, which produces a comprehensive dataset that includes important parameters like build time, resource utilisation, failure rate, latency, cost per build, security incidents, and user feedback. For accessibility, organisation, and a smooth interface with Google Looker Studio for further visualisation and analysis, this data is safely kept inside a Google Sheet. This process is illustrated in figure 3.1 below.

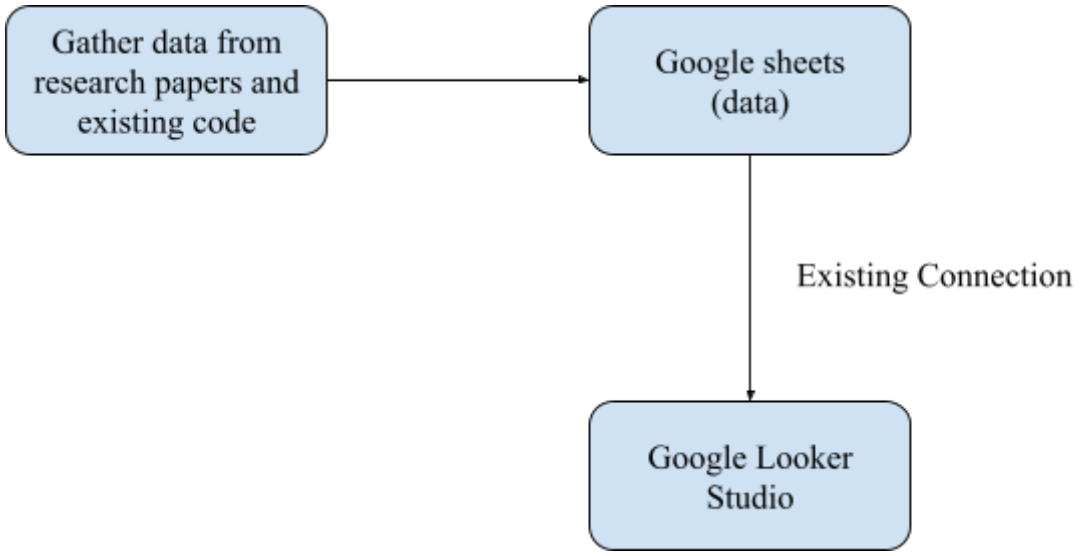

Figure 3.1: Flow of Data Collection and Analysis

We use a multimodal method to derive useful insights from the data. The fundamental tendency and variability of each metric across various configurations are summarised

and characterised by descriptive statistics, which include means, medians, standard deviations, and ranges. This analytical process is further strengthened by Google Looker Studio, which creates interactive dashboards and reports that effectively convey important findings through the use of stunning visuals. Heatmaps provide deeper insights into intricate linkages within the data, scatter plots show correlations between variables, bar charts enable performance comparisons between configurations, and line graphs visualise trends and patterns over time.

Our goal is to derive meaningful insights from the simulated dataset by giving priority to strong data collection and analysis approaches. As a result, it will be easier to create evidence-based suggestions for improving ABP testing and construction performance in subsequent studies and real-world applications.

3.6 Ethical concerns related to the research

While ABPs offer private networks efficiency, they also bring up ethical concerns that need to be addressed. This subsection delves into possible effects on biases in algorithms, job displacement, data privacy, and the sustainability of the environment. We aim to clear the way for the ethical, secure, and responsible development and application of ABPs in private networks by addressing these issues head-on (Rubert and Farias, 2022). Some of these issues might be:

Data security and privacy:

Firstly, automating network deployment procedures may lead to the creation of fresh points of entry and vulnerability for hackers. To reduce these risks, the research should take strong security precautions and penetration testing into account. Another concern would be the sensitive information about user behaviour and network activity may be gathered by the pipeline. The storage, security, and usage of this data must be considered in the research to ensure openness and adherence to data privacy laws. Lastly, cybercriminals may find it simpler to control a network or pilfer information when there is automation in place. To stop unwanted access and keep track of actions,

the project should take audit logs and access control systems into account (de Bruin and Floridi, 2017)

Fairness and Bias:

One major concern would be the automated pipeline may reinforce pre existing biases in the data it processes or the network design. In order to guarantee just and equitable network access and resource allocation, the research should take into account techniques for detecting and reducing algorithmic bias. (Soares *et al.*, 2022)

Opacity and accountability can also be categorised as an ethical issue. It can be challenging to comprehend and troubleshoot complex automated systems. By offering precise documentation and explainability procedures for the pipeline's decision-making procedures, the research should allay worries about transparency, (Elazhary *et al.*, 2022) illustrated in their article.

Impact on society and loss of employment:

Automation and job loss: IT workers may lose their jobs as a result of automating network deployment operations. In addition to examining strategies to lessen job displacement, such as retraining programmes or the creation of new work opportunities, the research should take into account any potential societal impacts.

Digital gap and accessibility: Automating network administration could make people and communities who don't have access to technology or training even more marginalised. The study should address the moral ramifications of growing the digital divide and provide ways to guarantee that everyone has fair access to network resources (Garg and Garg, 2019)

Control and supervision:

Absence of defined rules: Private network automated systems regulations and ethical frameworks are still developing. To guarantee the appropriate creation and implementation of such systems, the research ought to promote precise rules and industry best practices (Steffens, Lichter and Döring, 2018)

Accountability and transparency for algorithms: How to guarantee accountability and transparency for algorithmic decision-making, as well as who bears responsibility for automated systems' activities, should be the focus of this project (Kessel and Atkinson, 2018)

By anticipating these ethical challenges, research on automated build pipelines can assure responsible development and deployment of these technologies in the future, encouraging secure, fair, and sustainable private networks.

3.7 Chapter Summary

The design for the proposed model was discussed in this chapter. It provides a thorough justification for the selected approach, demonstrating how it supports the project's objectives. The chapter then dives into the data that was used, describing its properties and the methods used to collect and examine it. Transparency and ethical research techniques are ensured by addressing ethical issues pertaining to simulated data. This chapter essentially establishes the foundation for the data-driven insights that follow, making it easier to comprehend how different aspects affect the efficiency of the build pipeline.

CHAPTER 4 - DATA, EXPERIMENTS, AND IMPLEMENTATION

4.1 Appropriate modelling in relation to project

This study makes use of a specialised simulation model that is intended to replicate real-world building scenarios as precisely as possible and to enable controlled experimentation with ABP testing in the particular context of the project. Key characteristics and variables related to the project's particular building procedures, resource limitations, and security requirements are incorporated into the model.

The intricate linkages between ABP testing, build time, resource utilisation, failure rates, latency, cost per build, security events, and user feedback are precisely captured by this customised model. It makes it possible to precisely modify resource allocations, security measures, and ABP configurations, allowing for the monitoring of their effects on these crucial performance indicators. The extensive simulated data produced by this controlled environment precisely matches the project's goals and needs.

The model's parameters and variables are carefully calibrated to meet the particulars of the project in order to guarantee the model's relevance and applicability within it. As a result, in the particular context of the project, decisions about ABP optimization and construction performance enhancement will be made directly informed by the model's insights.

But it's important to recognise that every simulation model has its own set of constraints. It provides a useful controlled environment for research, but it is unable to perfectly capture the complexities and unpredictability of actual construction projects. Because of this, extending the model's results to actual situations should be done so cautiously, taking uncertainties and other outside influences into account.

We use a two-pronged method to solve these limitations and guarantee the continuous correctness and efficacy of the model. First, we continuously validate the model's outputs throughout the research process by contrasting it with real-world data that is readily available or with expert commentary. Second, we use an iterative refinement approach, making changes to the model's assumptions and parameters in response to the validation outcomes. This enhances the validity of the model's conclusions and guarantees that it will continue to be in line with the particular needs of the project

Through the strategic application of this customised simulation model, which is continually validated and improved, our goal is to produce trustworthy insights that directly inform the optimisation of ABP and performance in the particular context of the project. Through recognition of the inherent limitations of any simulation and implementation of suitable mitigation techniques, our goal is to produce solid results with immediate practical significance for the project's successful completion.

4.2 Techniques, algorithms, mechanisms

This section examines the particular methods, procedures, and systems used in the simulation model to replicate actual building situations and assess the effects of ABP testing.

We utilise an agent-based methodology in which discrete entities (agents) embody different aspects of the construction process (e.g., tasks, resources) communicate and adjust according to pre-established rules and algorithms. The intricate interactions between many components in a real-world construction project are mirrored in this dynamic agent interaction.

We also use Monte Carlo simulations to take into consideration the inherent uncertainty and randomness found in real-world building. By adding stochastic components to the model and producing several conceivable results for every experiment, this method offers probabilistic insights into the effects of ABP in different scenarios.

To simulate the sequential character of construction processes, we apply discrete event simulation. This enables the precise assessment of performance indicators like build time and resource utilisation by allowing us to follow the occurrence of separate events (e.g., task completion, resource allocation) over time.

Descriptive statistics, a technique employed in this study, entails summarising important metrics for various setups, including build time, resource utilisation, and failure rates. A summary of the data is given by descriptive statistics, making it possible to compare and spot patterns or trends. Data analysis requires not just descriptive statistics but also the use of visualisations. Data can be visually represented with charts, graphs, and dashboards made possible by programmes such as Google Data Studio. By assisting in the detection of correlations and patterns among data, these visualisations enable analysts to make well-informed judgements by drawing on new information.

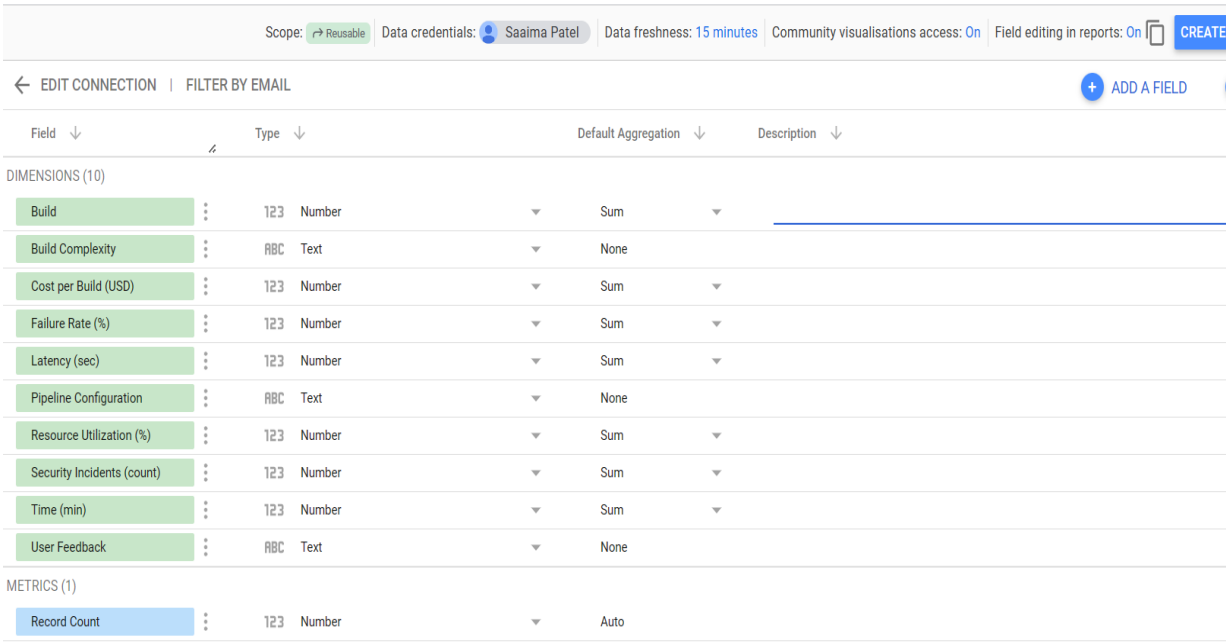

Figure 4.1: Visualisation of Data in Google Looker Studio

Configurations that simultaneously take into account variables like cost, speed, and security can be found through multi-objective optimization. Sensitivity analysis supports optimisation efforts by illuminating how modifications to particular parameters or settings impact build pipeline efficiency.

Apart from these methods and processes, additional factors can improve the study. In order to determine qualitative aspects of build pipeline performance and user satisfaction, user data and feedback are analysed. By weighing the trade-offs between various configurations in terms of cost and performance gains, cost-benefit analysis offers guidance for making decisions. To evaluate a configuration's vulnerability and find possible mitigations, security vulnerability analysis entails simulating security attacks on various setups.

These methods, procedures, and systems offer a foundation for putting the automated build pipeline paradigm into practice. The needs, architecture, and preferences of the private network under consideration will determine which tools and technologies are best.

4.3 Main functions of models or frameworks

Within our study methodology, we highlight the following essential functions, models, and frameworks to fulfil the research objectives: evaluating the impact of ABP testing on construction performance; and optimising ABP configurations for increased efficiency and security.

Simulation model: The main instrument for carrying out controlled tests and producing data on the interaction between ABP configurations and construction performance metrics is the customised simulation model, which combines agent-based, Monte Carlo, and discrete event simulation approaches.

Data visualisation and analysis: With interactive dashboards, line graphs, bar charts, scatter plots, and heatmaps, Google Looker Studio facilitates the investigation and dissemination of insights from the simulated data.

Error generation was added to the simulated data process in order to assess how reliable ABP configurations are. This module presents plausible faults and malfunctions, enabling evaluation of the effects these mistakes have on the system. To assess the effectiveness of different security measures inside ABP settings and provide insights into potential weaknesses and areas for improvement, a module for simulating security vulnerabilities was also introduced. This guarantees accurate outcomes and aids in locating any bottlenecks that can impede the automated procedure.

Finding the best ABP configurations is made possible by model experimentation, correlation analysis, and data visualisation. These methods help find ABP configurations that optimise productivity, reduce mistake rates, and provide strong security.

The primary purpose of the suggested model is to provide a variety of realistic build pipeline scenarios that illustrate the potential effectiveness of the installed ABP. It enables the investigation of actual situations in various combinations. It enables the depiction of trends or patterns via charts, for example. Additionally, the model enables evaluation of other characteristics or settings that could impact the ABP..

Through the strategic application of these tools and approaches, the research seeks to optimise ABP testing and building processes in order to produce actionable insights that improve project outcomes.

4.4 Chapter Summary

This chapter explores the models, simulations, and implementation techniques used for the proposed model, which form the basis of the research. It starts by outlining the selected models and how they relate to the goals of the project, emphasising how well-suited they are for assessing and enhancing build performance. Following that, the chapter delves further into the particular methods, techniques and workings of these models. The chapter then turns its attention to the fundamental features of the created models or frameworks. It describes the models' operation, the aspects of build performance they affect, and how they help the project reach its objectives. It concludes with a brief overview of the models selected and their relevance, as well as a brief summary of the mechanisms and approaches used.

CHAPTER 5 - RESULTS AND DISCUSSIONS

5.1 Results Presentation

The data was organised and collected in Google Sheets, and then it was connected to Google Looker Studio for additional analysis and visualisation. A number of charts were produced, with the creation of scatter charts being the primary focus at first.

The scatter plots in figure 5.1 offered insightful information on the correlations between various parameters, especially with regard to the cost per build

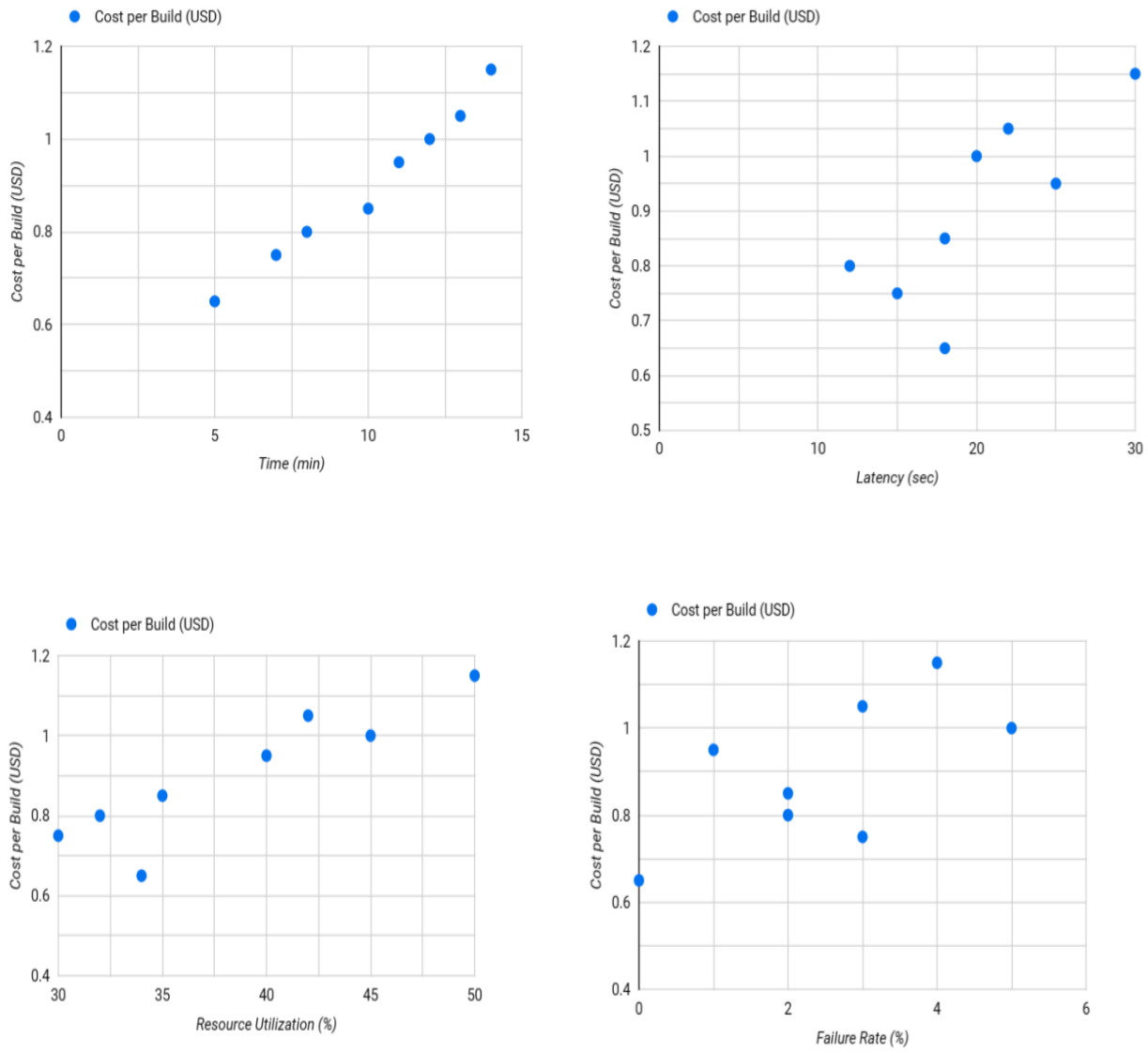

Figure 5.1: Relationship Analysis: Cost per Build and Operational Metrics

Positive correlations typically indicate that when one variable rises, the other variable is also likely to rise. This implies a clear connection between the variables. Conversely, a low or nonexistent correlation suggests that the variables have little to no association with one another. From figure 5.1, we can derive:

- Given the positive association between time and cost per build, it may be inferred that build times tend to increase along with build costs. This suggests that longer build timeframes could necessitate inefficiency or the need for more resources.
- Changes in latency do not appear to have a major effect on the cost per construction, as seen by the lack of a clear trend or link between latency and cost per build. This implies that changes in latency could not have a direct impact on the build process' total cost.
- Higher resource utilisation usually translates into higher build costs, according to the positive association found between resource utilisation and cost per build. This may suggest that reducing the cost per build can be achieved by assigning more resources or by employing resources more effectively.
- It appears that changes in failure rate have no effect on cost per construction because there is no pattern or link between failure rate and cost per build. This suggests that build-related errors might not have a direct impact on the final cost.

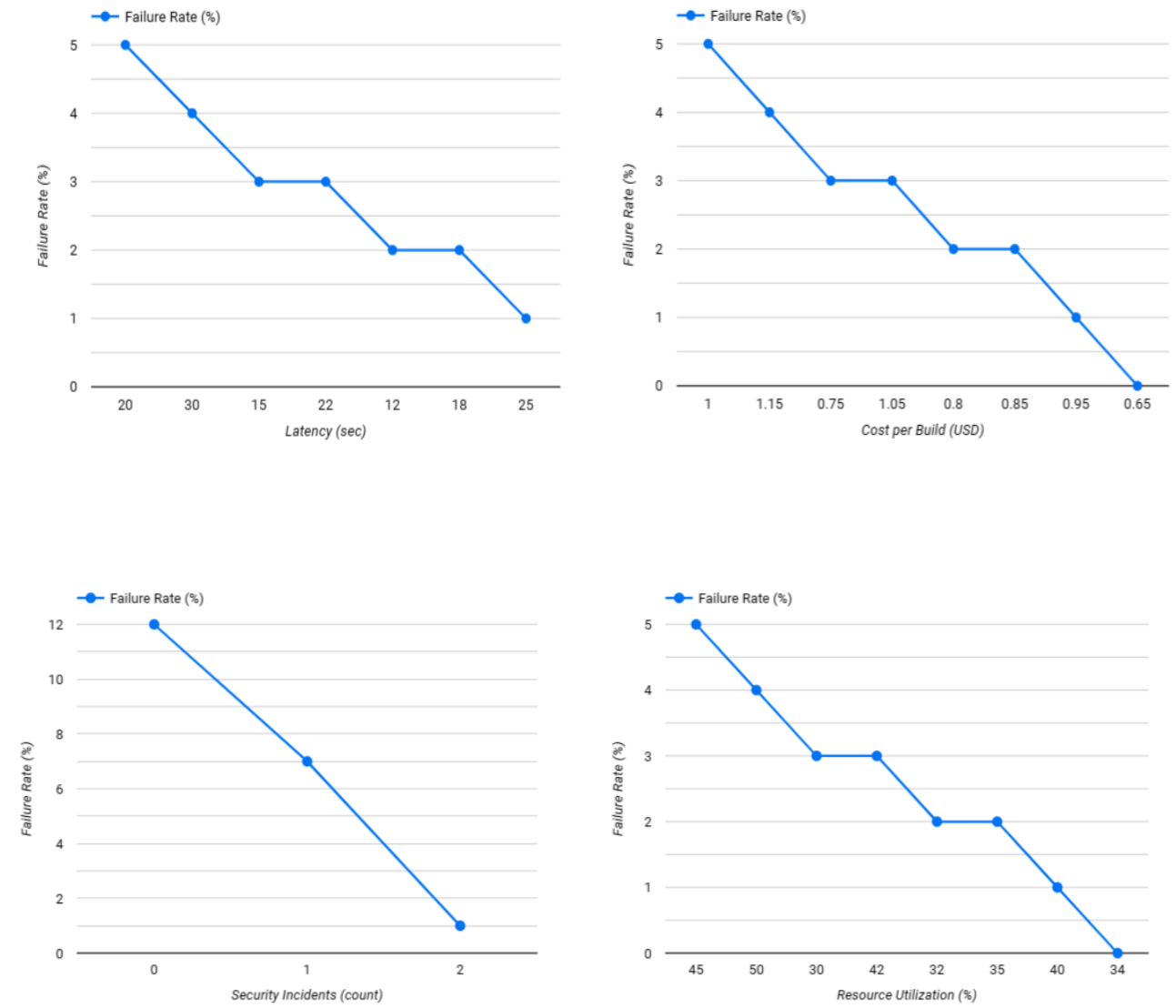

Figure 5.2: Relationship Analysis: Failure Rate and Operational Metrics

Figure 5.2 above shows line graphs with various operational metrics plotted against the failure rate. Based on these graphs, we can derive:

- The first graph looks at the connection between failure rate and latency (measured in seconds). It clearly displays a declining trend, meaning that the failure rate falls as delay rises. With a 20-second latency at the beginning of the graph, the failure rate is

approximately 5%. On the other hand, the failure rate falls to little over 1% as the latency rises to 25 seconds. This implies that a lower failure rate is caused by longer latency times.

- The second graph shows the correlation between the failure rate and the cost per build (in USD). It shows a downward tendency as well, much like the first graph. Failure rate drops in tandem with decreasing cost each build. For a build cost of one USD, the first data point indicates a failure rate of roughly 5%. But the failure rate falls below 1% when the cost per build drops to 0.65 USD. This implies that a lower failure rate is linked to lower costs per build.
- The third graph, which looks at the connection between the quantity of security incidents and the failure rate, is next. This graph shows a sharp downward trend, meaning that the failure rate sharply drops as the number of security events rises. When there are no security events at the beginning of the graph, the failure rate is somewhat less than 12%. Nevertheless, the failure rate drops to 0% when there are two security incidents. This implies that a significant drop in the failure rate may result from a greater attention on security issues.
- The fourth graph investigates the connection between the failure rate and resource utilisation (shown as a percentage). Similar to the earlier graphs, this one shows a declining trend. There is a decrease in both the failure rate and resource utilisation. With a resource utilisation of 45% at the start of the graph, the failure rate is marginally less than 5%. On the other hand, the failure rate falls to little over 1% as the resource utilisation falls to 34%. This suggests that maximising the use of available resources can result in a decreased failure rate.
- The graphs demonstrate that failure rate drops in tandem with decreases in latency, cost per build, security events, and resource utilisation. These results emphasise how crucial it is to optimise these factors in order to reduce the likelihood of failures. Furthermore, the failure rate decreases most noticeably when security incidents rise, according to the graph showing security incidents vs. failure rate.

5.2 Analysis of Results

Even though our build performance data at first glance showed some intriguing correlations, further investigation is needed to get useful information for cost optimisation.

Let us examine the results of figure 5.1 in order to gain a more thorough grasp:

- The fact that time and cost per build are positively correlated implies that higher expenses could be the outcome of longer build timeframes. It would help to investigate this link further by finding the ideal time frame for cost reduction without sacrificing the effectiveness or quality of the building process.
- It appears that variations in latency have no effect on the cost per construction because there is no discernible pattern or link between latency and cost per build.
- It is implied that greater resource utilisation results in higher costs by the positive link between resource utilisation and cost per build. This research implies that in order to reduce expenses while preserving appropriate levels of resource utilisation, careful resource management and optimisation techniques are required.
- It appears that changes in failure rate have no effect on cost per construction because there is no pattern or link between failure rate and cost per build. This research suggests that higher expenses might not always be directly caused by build process errors.

Based on the results obtained from figure 5.2, these are the results we can reach:

- The research indicates that, in the context of the data under consideration, a number of factors may have an impact on the failure rate based on the aforementioned results. The failure rate on the y-axis and the variables on the x-axis show a constant negative association, as seen in the line graphs.
- The failure rate falls with increasing latency, according to the study of the latency vs. failure rate graph. This suggests that greater latency durations for systems or processes are associated with a reduced failure rate. It can be deduced that giving oneself enough time to process or respond lowers the probability of errors.
- The failure rate likewise falls as the cost per build does, according to the link between the two variables. This shows that a reduced failure rate may be achieved by optimising expenses and resource allocation during the build phase. It suggests that fewer failures are linked to projects that are more economical and successful.
- There is a strong correlation, according to the failure rate graph and security incident analysis. The failure rate falls as the number of security incidents rises. This result implies that a reduced failure rate is a result of a greater emphasis on security issues.

It suggests that strong security protocols and incident handling can successfully reduce malfunctions

- Finally, the resource utilisation vs. failure rate study shows that a decrease in resource utilisation is accompanied with a decrease in the failure rate. This implies that maximising the use of available resources may result in a decreased failure rate. It suggests that minimising failures requires effective resource allocation and management.
- All things considered, the analysis emphasises how important different factors are in determining the failure rate. It implies that reducing failure rates can be achieved by optimising resource usage, security incidents, latency, and cost per build. These results shed light on possible areas where systems or procedures should be strengthened in order to lower failure rates and increase overall performance.

5.3 Comparison to Related Work

Research gaps are filled in a number of areas, including cost effectiveness, handling merging conflicts, semantic versioning, handling failed builds, automated build from git provider, and private network applicability, as mentioned in table 2.1

The analysis emphasises the connection between failure rates and cost per build in terms of cost effectiveness. It highlights the significance of optimising build costs for better performance by showing that a lower cost per build is linked to a lower failure rate.

It also shed light on cost-effective methods that reduce failures by examining the connection between cost per build and failure rates. But the study doesn't concentrate on identifying the underlying reasons behind unsuccessful builds and creating workarounds for them. This might be useful in determining how to handle builds that fail.

(Garg and Garg, 2019), is an example of how it does not provide a very cost effective approach. It uses Gitlab which takes up to $50 per month. Furthermore, the personnel require their own fees too. With the proposed model, if implemented as a package, would simply require connecting the package, and following instructions as per the package guidelines. The costs incurred would be for cloud hosting and git minutes on github.

5.4 Implications of Results

In this section, a summary of the consequences of the findings based on the previous analysis will be carried out.

Based on the analysis of figure 5.1, these are the implications:

- The fact that latency and cost per construction do not significantly correlate suggests that changes in latency might not have an immediate effect on total costs. This implies that enterprises might not have to concentrate their efforts on cutting latency only on financial concerns.
- Greater resource utilisation typically translates into greater expenses, as seen by the positive association found between resource utilisation and cost per build. Organisations should concentrate on efficiently assigning and managing resources to reduce waste and guarantee effective utilisation in order to optimise costs.
- There may not be a direct link between construction process failures and higher expenses, as evidenced by the weak connection found between failure rate and cost per build. This suggests that rather than focusing on cost reduction, organisations should prioritise efforts to eliminate failures largely for reasons related to quality and reliability.
- Overall, the data points to a direct correlation between cost reduction and build time and resource optimisation. It is crucial to remember that these implications are predicated on the examination of particular variables and may change based on the circumstances and particulars of the build process.

The implications of the results from figure 5.2 are:

- The research identifies important factors that can be adjusted to enhance system functionality and lower failure rates. Through the implementation of strategies such as latency reduction, cost minimization per build, security incident handling, and resource optimisation, enterprises can improve their overall efficiency and performance.
- Organisations can prioritise and bolster their security procedures by comprehending the connection between failure rates and security events. Organisations may reduce

risks and minimise failures by investing in strong security standards, incident management systems, and proactive security practices.

- According to the data, lowering the cost of each build can result in a lower failure rate. This conclusion suggests that in order to reduce failures while preserving operational efficiency, organisations can adopt cost-effective techniques, optimise their resource allocation, and streamline their operations.
- The investigation sheds light on the variables affecting failure rates. Systems and procedures can be continuously monitored, analysed, and improved with the use of this data. Through ongoing assessment and optimisation of latency, expenses, security protocols, and resource allocation, establishments can pursue ongoing enhancement and reduce failure frequencies.
- The outcomes provide insightful data to aid in decision-making. This study can help organisations make well-informed decisions about budgeting, resource allocation, system design, and security measures. Organisations can make better and more efficient decisions if they align their choices with the objective of lowering failure rates.
- Reduced failure rates could increase customer satisfaction. By optimising system performance, decreasing downtime, and averting malfunctions, companies would be able provide their clients a more dependable and fulfilling experience. Positive brand perception and enhanced client loyalty may follow from this.

5.5 Chapter Summary

The first section of this chapter presents the major findings, demonstrating the findings. After that, it explores deeply into the analysis of the findings, exposing the insights and hidden patterns found in the data. Afterward, a comparison is made between the acquired results and current research and industry standards. This uses the comparison made in table 2.1 earlier. The chapter comes to a close with a discussion of the findings' wider ramifications. It examines the possible advantages and uses of the results.

CHAPTER 6 - SUMMARY AND CONCLUSION

6.1 Summary of Main Findings

The results of the investigation show that while variations in latency may not directly affect the final cost, reducing build time can result in cost savings. Costs often rise with higher resource utilisation, and quality and dependability are largely benefited by lower failure rates. Cost reduction can be immediately impacted by optimising build time and resource utilisation, while more research and validation are advised for decision-making tailored to the particular circumstance.

The correlations between many variables, including time, cost per build, delay, resource utilisation, and failure rate, are illustrated in Figure 5.2. It comes to the conclusion that time and cost per build are positively correlated, indicating that longer build periods can call for more resources. Nevertheless, there is no discernible relationship between latency and cost per build, suggesting that variations in latency would not have a big impact on the total cost. Additionally, it observes a positive link between cost per build and resource utilisation, suggesting that higher resource utilisation results in higher expenses. Failed builds, however, do not appear to be correlated with cost per build, suggesting that the cost may not be directly affected by build failures.

The graphs in figure 5.2 demonstrate how the failure rate falls in tandem with decreases in latency, cost per build, security events, and resource utilisation. To reduce failures, it's critical to optimise these variables. The failure rate significantly decreases as security incidents rise, according to the security incidents graph. Additionally, it draws attention to a negative association between the failure rate and factors like latency, cost per build, security issues, and resource utilisation. It implies that a decreased failure rate can be attained by longer latency periods, reduced build costs, increased focus on security incidents, and optimised resource use. The results shed light on possible areas for development in order to lower failure rates and improve system performance as a whole.

6.2 Attainment of Research Objectives

Chapters 1 and 2 both provided a thorough identification and analysis of the dangers and problems related to the implementation of automated build pipelines in private networks. The

research offered a thorough understanding of the various roadblocks and vulnerabilities that may occur during the implementation process through thorough analysis and a review of the literature.

Using this foundation as a starting point, the research went on to create an extensive framework/model for evaluating, organising, and carrying out the deployment of an automated build pipeline made especially for private networks. In Chapter 2, the first framework was carefully designed with the risks and challenges that were recognised in mind. This framework functioned as a road map to direct the research's following development and execution stages.

The research made use of Google Looker Studio and Sheets' features to make the framework come to life. The automated build pipeline concept was developed and put into practice using these platforms in a private network setting. The project sought to minimise delay in the build pipeline process, handle semantic versioning properly, and gain operational efficiency by utilising these technologies.

After the implementation stage, the research carefully examined and assessed the outcomes derived from the created model. A thorough evaluation was carried out to ascertain the degree to which the goals were achieved. This assessment covered a number of topics, including the automated build pipeline's throughput, compliance with semantic versioning guidelines, effectiveness in resolving merging disputes, and decrease in build failures.

By means of this thorough examination and assessment, the study aimed to determine whether the research goals had been met. A definitive conclusion about the accomplishment of the goals could be reached by closely examining the data, pointing out any gaps or limits, and contrasting them with the intended results.

In conclusion, the study process includes a methodical approach to problem identification, framework development, model implementation, and outcome evaluation. By going through this procedure, the study sought to meet the stated goals and advance our understanding of how to deploy automated build pipelines in private networks.

6.3 Contribution to the body of knowledge

The study's research has significantly added to our understanding of the subject of automating build pipeline implementation on private networks. These are the principal contributions:

The hazards and difficulties of deploying automated build pipelines in private networks were noted and thoroughly examined in the study. The research adds to the body of knowledge by elucidating the various traps and weaknesses that organisations could run into during the implementation process and by offering a thorough comprehension of these barriers.

For the purpose of evaluating, organising, and carrying out the deployment of an automated build pipeline designed especially for private networks, a framework or model was created. For enterprises looking to implement automated build pipelines in their private network settings, this framework is an invaluable tool. The created model offers recommendations and best practices to guarantee operational effectiveness, manage semantic versioning skillfully, and reduce build pipeline downtime.

The study illustrated how the automated build pipeline model could be developed and implemented in private networks by using tools like Google Sheets and Google Looker Studio in a realistic manner. The research advances the practical knowledge and comprehension of deploying automated build pipelines utilising widely accessible platforms by demonstrating the use of these tools.

The research evaluated the efficacy and performance of the automated build pipeline model through a thorough examination and analysis. Through the assessment of factors including build failure reduction, merging dispute resolution, semantic versioning adherence, and operational efficiency, the study offers significant understanding of the effectiveness and influence of the created model.

To sum up, this research study adds a great deal to the body of knowledge by defining problems, creating a framework or model, demonstrating real-world application, assessing findings, and offering helpful suggestions. Through tackling these facets, the study contributes to the comprehension and expertise of deploying automated build pipelines in

private networks, aiding establishments in their endeavours to enhance operational effectiveness and reduce build failures.

6.4 Challenges and Limitations faced

One of the challenges that arose during this research was the financial implications of using specific tools and services, such Google Cloud Services or comparable platforms. These cloud-based services frequently include usage or membership costs, which can put a strain on finances. The size of the infrastructure, the particular features and functions needed, and even the length of time can all affect how much these products and services cost. Moreover, costs for continuing support and maintenance were taken into account. The total cost of adopting and maintaining the automated build pipeline may be further impacted by extra costs for upgrades, licensing, and technical support after the initial setup.

The compatibility and processing capacity constraints of my current computer equipment presented another challenge when thinking through the project. My computer systems would not work with the particular platforms or tools needed for the automated build pipeline, so compatibility problems would occur. This involved problems like out-of-date software versions, unsupported operating systems, or hardware constraints that hinder the pipeline's constituent parts from performing as intended.

Moreover, another constraint was the inadequate processing capacity of my personal computers. Significant processing power and memory are needed to handle huge codebases, run resource-intensive build procedures, and execute several builds at once.

6.5 Future works

The following methods, procedures, and systems can be taken into consideration for putting the automated build pipeline model for private networks into practice within the framework of the project. Below is an outline of how the implementation of the system would work:

1.) Tools for Continuous Deployment and Integration (CI/CD): Build pipeline operations can be automated by utilising a variety of CI/CD solutions. Code compilation, testing, and deployment can be automated with well-known tools like Jenkins, GitLab CI/CD,

or Travis CI. A multitude of built-in methods and algorithms are usually available in these programmes to help automate and optimise the build process.

2.) Version control systems, such as Git, are useful for tracking revisions, managing code changes, and maintaining the integrity of the codebase. The utilisation of mechanisms like branching, merging, and pull requests can promote developer collaboration and guarantee a seamless and regulated integration process.

3.) Automated Testing Frameworks: The testing stage of the build process can be automated with the use of automated testing frameworks like Selenium for web apps and JUnit for Java. These frameworks offer tools and techniques for creating and running automated tests, producing test reports, and identifying problems or regressions in the source code.

4.) Containerisation: Applications can be containerised and their deployment can be orchestrated with the help of technologies such as Docker and Kubernetes. Applications and their dependencies can be packaged via containerisation, which ensures consistency in various settings. Kubernetes orchestration technologies facilitate effective containerised application scaling, load balancing, and management.

5.) Security Mechanisms: To safeguard private networks, security measures must be put in place at every stage of the build process. To find and fix security flaws, methods like vulnerability scanning, penetration testing, and code analysis tools (like SonarQube) can be used. To safeguard data privacy and prevent unwanted access, other measures include access control systems, secure communication protocols, and encryption methods.

6.) Monitoring and Logging Mechanisms: By putting them in place, it will be easier to keep track of the build pipeline's health and performance. Proactively identifying problems and performance bottlenecks is made possible by the collection and analysis of logs, metrics, and events using tools like Prometheus and the ELK (Elasticsearch, Logstash, Kibana) stack.

6.6 Chapter Summary

The key findings are thoroughly summarised in this chapter. The chapter also provides a critical assessment of the research's performance in meeting its original goals, emphasising the degree to which the objectives were fulfilled and the advances made in the field of build pipeline optimisation. The research's contribution to the body of current knowledge is then

addressed. It illustrates how the discoveries deepen our understanding of build pipeline optimisation and may provide new avenues for research in the future. It also recognises the difficulties and constraints that arose throughout the investigation. It ends with suggesting directions for further research that could be motivated by the findings.

REFERENCES

Arachchi, S.A.I.B.S. and Perera, I. (2018) 'Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management', in *2018 Moratuwa Engineering Research Conference (MERCon)*. *2018 Moratuwa Engineering Research Conference (MERCon)*, pp. 156–161. Available at: https://doi.org/10.1109/MERCon.2018.8421965.

Bello, Y. *et al.* (2022) 'Continuous Integration and Continuous Delivery Framework for SDS', in *2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. *2022 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 406–410. Available at: https://doi.org/10.1109/CCECE49351.2022.9918437.

de Bruin, B. and Floridi, L. (2017) 'The Ethics of Cloud Computing', *Science and Engineering Ethics*, 23(1), pp. 21–39. Available at: https://doi.org/10.1007/s11948-016-9759-0.

Debroy, V. and Miller, S. (2020) 'Overcoming Challenges With Continuous Integration and Deployment Pipelines: An Experience Report From a Small Company', *IEEE Software*, 37(3), pp. 21–29. Available at: https://doi.org/10.1109/MS.2019.2947004.

Dwivedi, Y.S., Semalty, G. and Moondra, A. (2021) 'Predictive Technique To Improve Classification On Continuous System Deployment', in *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*. *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pp. 1–4. Available at: https://doi.org/10.1109/CONECCT52877.2021.9622682.

Elazhary, O. *et al.* (2022) 'Uncovering the Benefits and Challenges of Continuous Integration Practices', *IEEE Transactions on Software Engineering*, 48(7), pp. 2570–2583. Available at: https://doi.org/10.1109/TSE.2021.3064953.

Gallaba, K. (2019) 'Improving the Robustness and Efficiency of Continuous Integration and Deployment', in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 619–623. Available at: https://doi.org/10.1109/ICSME.2019.00099.

Garg, Somya and Garg, Satvik (2019) 'Automated Cloud Infrastructure, Continuous Integration and Continuous Delivery using Docker with Robust Container Security', in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 467–470. Available at: https://doi.org/10.1109/MIPR.2019.00094.

Hung, P.D. and Giang, D.T. (2019) ‘Continuous Integration for Android Application Development and Training’, in *Proceedings of the 3rd International Conference on Education and Multimedia Technology*. New York, NY, USA: Association for Computing Machinery (ICEMT ’19), pp. 145–149. Available at: https://doi.org/10.1145/3345120.3345158.

Jin, X. and Servant, F. (2020) ‘A cost-efficient approach to building in continuous integration’, in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. New York, NY, USA: Association for Computing Machinery (ICSE ’20), pp. 13–25. Available at: https://doi.org/10.1145/3377811.3380437.

Kessel, M. and Atkinson, C. (2018) ‘Integrating Reuse into the Rapid, Continuous Software Engineering Cycle through Test-Driven Search’, in *2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE). 2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE)*, pp. 8–11. Available at: https://ieeexplore.ieee.org/document/8452100.

Koopman, M. (2019) *A framework for detecting and preventing security vulnerabilities in continuous integration/continuous delivery pipelines*. University of Twente. Available at: http://essay.utwente.nl/78048/

Lunde, B.A. and Colomo-Palacios, R. (2020) ‘Continuous practices and technical debt: a systematic literature review’, in *2020 20th International Conference on Computational Science and Its Applications (ICCSA). 2020 20th International Conference on Computational Science and Its Applications (ICCSA)*, pp. 40–44. Available at: https://doi.org/10.1109/ICCSA50381.2020.00018.

Mysari, S. and Bejgam, V. (2020) ‘Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible’, in *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE). 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pp. 1–4. Available at: https://doi.org/10.1109/ic-ETITE47903.2020.239.

Pinto, G. *et al.* (2018) ‘Work practices and challenges in continuous integration: A survey with Travis CI users’, *Software: Practice and Experience*, 48(12), pp. 2223–2236. Available at: https://doi.org/10.1002/spe.2637.

Railić, N. and Savić, M. (2021) ‘Architecting Continuous Integration and Continuous Deployment for Microservice Architecture’, in *2021 20th International Symposium INFOTEH-JAHORINA (INFOTEH). 2021 20th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pp. 1–5. Available at: https://doi.org/10.1109/INFOTEH51037.2021.9400696.

Riti, P. (2018) *Pro DevOps with Google Cloud Platform With Docker, Jenkins, and Kubernetes*. 1st ed. 2018. Berkeley, CA: Apress. Available at: https://doi.org/10.1007/978-1-4842-3897-4.

Rubert, M. and Farias, K. (2022) 'On the effects of continuous delivery on code quality: A case study in industry', *Computer Standards & Interfaces*, 81, p. 103588. Available at: https://doi.org/10.1016/j.csi.2021.103588.

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*, PP. Available at: https://doi.org/10.1109/ACCESS.2017.2685629.

Soares, E. *et al.* (2022) 'The effects of continuous integration on software development: a systematic literature review', *Empirical Software Engineering*, 27(3), p. 78. Available at: https://doi.org/10.1007/s10664-021-10114-1.

Steffens, A., Lichter, H. and Döring, J.S. (2018) 'Designing a Next-Generation Continuous Software Delivery System: Concepts and Architecture', in *2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE)*. *2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE)*, pp. 1–7. Available at: https://ieeexplore.ieee.org/document/8452099

Zampetti, F. *et al.* (2020) 'An empirical characterization of bad practices in continuous integration', *Empirical Software Engineering*, 25(2), pp. 1095–1135. Available at: https://doi.org/10.1007/s10664-019-09785-8.