SAVONIA

University of Applied Sciences

THESIS – BACHELOR'S DEGREE PROGRAMME
TECHNOLOGY, COMMUNICATION AND TRANSPORT

AUTOMATED DEVOPS WORKFLOWS FOR MANAGING SOFTWARE DEVELOPMENT PROJECTS

AUTHOR Habeebullah Lawal

PREFACE

First and foremost, all praises to Almighty Allah, Subhanahu wa ta'ala (SWT), who has guided me through my studies at Savonia University of Applied Sciences. It is with His grace that I have been able to pursue my academic goals and complete this thesis.

I would like to express my sincere appreciation to my thesis supervisor, Jussi Koistinen, whose expertise and insights have been invaluable throughout the research and writing process. His guidance and support have been pivotal in shaping this academic endeavor.

My gratitude also goes to my work supervisor, Antti Lappalainen, alongside Juha Kokora and Jukka Vidgren, who have provided significant guidance and support during my tenure at Ponsse Plc. Their contributions have been instrumental in aligning this thesis with practical industry standards and expectations.

I am immensely thankful to my teachers at Savonia, particularly Markku Kellomäki, Rajeev Kanth, and Teemu Matilainen, amongst others, who have enriched my learning experience and academic growth. Their teachings have been foundational in my educational journey. My classmates deserve special mention for their camaraderie and the collaborative spirit that made our learning environment both challenging and rewarding.

A heartfelt thanks to my DevOps team members, Pekka Ronkainen and Sami Juvani, the Digital Professional Student OGs, and all my other colleagues at Ponsse. Their collaboration and the dynamic work environment they foster have greatly contributed to my professional development and the completion of this thesis.

Finally, I must acknowledge the unwavering support and encouragement of my family and friends. Their constant encouragement and belief in my abilities have been a source of strength and motivation for me. Their sacrifices have not gone unnoticed, and I am forever grateful for their unwavering love and support.

This thesis represents not just an academic achievement but also a testament to the collective effort of many individuals who have contributed to my development both professionally and personally. Thank you all for being part of this journey.

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology, Internet of Things	
Author(s) Habeebullah Lawal	
Title of Thesis Automated DevOps Workflows for Managing Software Development Projects	
Date 7 June 2024	Pages/Number of appendices 43/0
Client Organisation /Partners Ponsse Plc	

Abstract

The challenges in the software development lifecycle at Ponsse Plc were identified through direct experience of the author while working within the company. These primarily involved the manual initiation and management of software projects, which were inefficient and often resulted in only partial compliance with the dynamic DevOps and IT specifications of the company. The main objective of this thesis was to automate the initiation process for software development projects, ensuring they align with Ponsse's standards, and to establish a structured system for documenting projects and managing access rights.

An automated project generation system was developed to address these challenges, integrating directly with Ponsse's existing IT infrastructure. The method applied involved designing automated workflows to replace manual processes and employing scripting for task automation along with API integrations for system interactions. This approach aimed to streamline project setups and enforce consistent project documentation and access control procedures, building on firsthand observations and intrinsic company knowledge.

The results of the implementation were mixed, with certain components functioning effectively and others not meeting the expected outcomes. While the automation of some processes led to improvements in project setup times and documentation practices, difficulties in fully integrating the system with all required IT systems and services were evident. These outcomes highlight the complexities involved in automating multifaceted IT operations and underscore the need for ongoing development to refine the automation system. Further studies are suggested to enhance the integration capabilities of the system and to explore adaptive solutions that can better accommodate the evolving specifications of DevOps environments.

Keywords
DevOps, CI/CD, IT Automation, Software Development, Workflow, Pipeline, Access Control, Infrastructure as Code

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

In the rapidly evolving landscape of software development, the adoption of DevOps practices has become increasingly crucial for organizations aiming to enhance operational efficiency and improve collaboration between development and operations teams. DevOps is not merely a set of practices but a culture that promotes the automation of processes and the seamless integration of various stages of software development, from design through development to production support. As more organizations embrace these methodologies, the need for advanced tools and systems to efficiently manage software projects becomes paramount.

This thesis focuses on a project initiated at Ponsse Plc, a leader in its industry, which has recognized the necessity of automating and streamlining its software development lifecycle to maintain competitive advantage and adapt to continuous technological changes. The overarching goal is to transform how software projects are initiated, managed, and documented, aligning them with the best practices in DevOps and IT management. By building on the foundation laid by previous initiatives and leveraging modern technologies, this thesis aims to address the gaps and enhance the efficacy of software project management at Ponsse.

Following this introduction, there are multiple subsections that detail the background and motivation for the project, clearly defined research objectives, and the scope and limitations of the study. These components collectively establish the framework for the research and development activities described and evaluated in subsequent sections of this document.

1.1 Background and Motivation

The project undertaken in this thesis builds on an existing initiative at Ponsse Plc, named the Ponsse DevOps Project Generator, which was originally designed to automate the process of software development project initialization according to Ponsse's DevOps and IT specifications. Despite its ambitious goals, development on that project has stalled, and it has failed to fully achieve its objectives. Currently, software development projects at Ponsse are still being created and initialized manually, relying heavily on 'copy and paste' actions. These manual processes only partially comply with Ponsse’s evolving DevOps and IT specifications, which are continuously changing and require a new approach that is modular, adaptable, and versatile enough to conform to these dynamic specifications.

Another significant reason for embarking on this thesis is the absence of a structured method for documenting information and configurations about software development projects within Ponsse. This lack of systematic documentation leads to inefficiencies and inconsistencies in managing project information across the organization. Furthermore, the management of access rights to resources needed in the software development lifecycle is poorly handled. These challenges in access management often result in bottlenecks and security concerns that can hinder the development process and affect overall project delivery.

This thesis aims to address these critical gaps by developing and implementing an automated system for generating, initializing, and managing Ponsse’s software development projects. This system

will ensure that projects not only comply with Ponsse's DevOps and IT specifications but also improve the efficiency and security of project setup and management. The focus of this thesis will be on establishing a foundation for an automation system tailored specifically to the management of a single page application (SPA), setting a precedent for future, more complex DevOps-oriented projects management systems within Ponsse's Digital Services and IT department.

1.2 Research Objectives

The main objective of this thesis is to design, develop, and implement an automated system that significantly enhances the generation, initialization, and management of software development projects at Ponsse Plc. This system aims to streamline the entire project lifecycle, from setup to deployment, ensuring full compliance with Ponsse's DevOps and IT specifications. Specific objectives of this thesis include:

- **Development of an Automated Project Generator**: Create a robust system that automates the creation and setup of new software projects, particularly addressing the complexities associated with initiating single page applications (SPA). This system will automate tasks that are currently performed manually, such as the copying of base code and configuration files, thereby reducing human error and increasing efficiency.
- **Implementation of Structured Documentation**: Establish a methodical approach to document project configurations and information systematically. This will provide a unified format and repository for all project-related data, facilitating easier access and management, enhancing transparency, and ensuring that all team members have up-to-date information.
- **Enhancement of Access Rights Management**: Design and integrate a comprehensive access control system within the project generator that manages and automates the assignment of appropriate access rights for project resources across various stages of the development lifecycle. This system will ensure that only authorized personnel have access to specific tools and resources, enhancing security and compliance with internal policies.
- **Adaptation to Dynamic Specifications**: Ensure that the automation system is flexible and adaptable enough to easily accommodate changes in Ponsse's DevOps and IT specifications without requiring extensive manual reconfiguration. This involves creating a modular system where components can be updated or replaced as needed to align with evolving organizational standards.

By achieving these objectives, the thesis will not only address the current inefficiencies and limitations identified in Ponsse's project management practices but also provide a scalable and adaptable framework that can serve as a model for future enhancements and expansion into other areas of software development within the organization.

1.3 Scope and Limitations

The scope of this thesis is dedicated to the design, development, and establishment of an automated system specifically focused on the management of software development projects within

Ponsse Plc. This automation system integrates various services and components into cohesive processes and workflows, adhering strictly to the specifications set by Ponsse's DevOps and IT guidelines. The system is designed to be a foundational tool for the structured and efficient management of software projects, particularly addressing the need for consistent documentation and effective access rights management.

In this thesis, the implementation will concentrate on the creation and management of a single page application (SPA). This specific focus serves as a proof of concept that will illustrate the feasibility and benefits of the automation system, setting the groundwork for future expansion to more complex, DevOps-oriented projects management systems. The development of this initial system aims to demonstrate significant improvements in project setup efficiency, configuration management, and security protocols compared to the current manual processes.

The limitations of the thesis are as follows:

- **Single Workflow Implementation:** Due to the constraints of time and resources, this thesis will develop and implement only one workflow and process. This process will cater exclusively to the initialization and management of a single type of software project, specifically an SPA.
- **Scope of Automation:** The automation will cover specific aspects of project management, including the generation of project scaffolding, configuration of DevOps tools, and setting of access controls. It will not encompass all potential areas of project management or the broader aspects of continuous integration and continuous deployment that may be included in future systems.
- **Client Organization Focus:** The research and development efforts are tailored specifically to the Digital Services and IT department of Ponsse Plc. While findings and methodologies may be applicable to similar environments, they are developed with the context and specific needs of Ponsse in mind, which may limit their direct applicability to other organizations without adjustments.

The defined scope and these limitations are critical to maintaining a focused approach throughout this project, ensuring that the system developed is robust within its intended application area while laying a strong foundation for future scaling and adaptation in broader project management contexts.

2 TECHNOLOGIES DEFINITION

In modern software development, several key technologies have emerged to enhance the efficiency, scalability, and reliability of applications and computer infrastructure. These technologies form the foundation of DevOps practices, ensuring streamlined processes, automated workflows, and seamless integration between development and operational tasks. Understanding these technologies is crucial for leveraging their full potential in delivering robust and scalable software solutions. This section provides an overview of some of the essential technologies used in the practical implementation of the thesis.

2.1 Application Programming Interface (API)

Application Programming Interfaces (APIs) are a set of protocols, tools, and definitions in modern software development that enables communication and interaction between different software systems. An API defines a set of rules and protocols for how one software component can interact with another, allowing applications to access the functionality or data of other services or platforms without the need to understand the underlying code. (Postman s.a.)

According to Figure 1, APIs work by receiving requests from clients, processing them, and returning appropriate responses. This interaction typically follows a request-response model, where a client sends a request to the API endpoint, and the API server processes this request and sends back the required data or performs the requested action. APIs use standard protocols such as Hypertext Transfer Protocol/Hypertext Transfer Protocol Secure (HTTP/HTTPS) for communication and often return data in formats like JavaScript Object Notation (JSON) or Extensible Markup Language (XML).

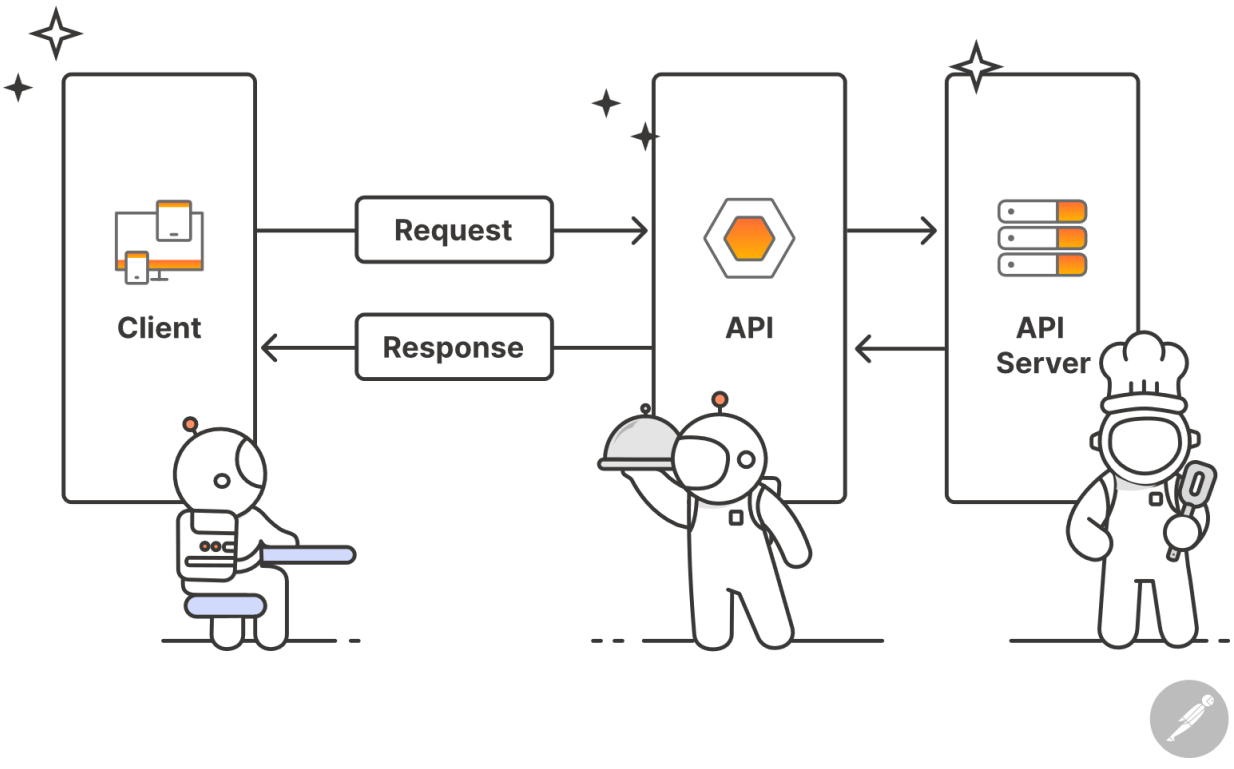

Figure 1. How APIs Work (Postman s.a)

2.2 Continuous Integration and Continuous Delivery/Continuous Deployment (CI/CD)

Continuous Integration (CI) and Continuous Delivery/Continuous Deployment (CD) are essential practices in modern software development, aimed at enhancing the efficiency, quality, and reliability of the software development lifecycle.

Continuous Integration (CI) is the practice of integrating code changes into a central shared repository frequently. Each integration is automatically verified by building the application and running automated tests to detect integration errors as early as possible. This frequent integration helps in identifying and addressing issues promptly, thereby reducing integration problems and ensuring a more stable codebase. (GitLab Inc. s.a.)

Continuous Delivery (CD) is the practice that extends the CI process by automating the release process. Continuous Delivery ensures that the codebase is always in a deployable state, enabling manual or automated (with continuous deployment) deployments to a production environment. (GitLab Inc. s.a.)

Continuous Deployment (CD) essentially enables automatic deployments of applications after the completion of the continuous delivery process. As there are no manual validation checks before deployments, continuous deployment depends on robust test automation and stringent measures to ensure that the application adheres strictly to specified standards. This automation reduces manual intervention, accelerates the release process, and ensures that new features, improvements, and bug fixes are delivered to users swiftly and consistently. (GitLab Inc. s.a; Red Hat, Inc. 2023.)

The implementation of CI/CD pipelines involves various stages, including source code management, automated testing, integration, deployment, and monitoring. These pipelines are integrated with version control systems and other DevOps tools to facilitate seamless and automated workflows. This integration ensures that every code change is systematically built, tested, and deployed, minimizing human errors and maximizing efficiency.

The benefits of CI/CD are manifold, including faster time-to-market, improved collaboration among development and operations teams, enhanced code quality, and the ability to quickly respond to market changes and customer feedback. By adopting CI/CD practices, organizations can achieve a more agile and resilient software development process, ensuring continuous improvement and innovation. Essentially, CI/CD practices are fundamental to modern software engineering, enabling rapid, reliable, and repeatable software delivery, thus playing a crucial role in the success of DevOps initiatives. (JetBrains s.r.o. s.a; BrowserStack 2023.)

2.3 Cloud Services

Cloud services refer to a wide range of services delivered over the internet, enabling on-demand access to computing resources without the need for low-level resources management by the user. Cloud services are generally offered through three standard models which are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). (Akamai Technologies, Inc. s.a; International Business Machines Corporation s.a.)

Infrastructure as a Service (IaaS) provides computing resources on demand, on a pay-as-you-go basis. IaaS offers significant flexibility and scalability, enabling organizations to quickly provision new resources, scale existing resources up or down according to demands without investing in physical hardware. IaaS also removes the need for maintenance as management and maintenance of the underlying infrastructure is handled by the cloud service provider. (Microsoft Corporation s.a.)

Platform as a Service (PaaS) offers a complete development and deployment environment in the cloud, with resources that allow applications to be built, tested, deployed, and managed. PaaS solutions which often include computing infrastructure alongside middleware, development tools, etc. are designed to support the entire application lifecycle, removing the need to manage the underlying infrastructure. (Microsoft Corporation s.a.)

Software as a Service (SaaS) delivers cloud-based software applications and solutions over the internet on a pay-as-you-go basis. The software can be accessed over the internet via a web browser, eliminating the need for installation, maintenance, and management of the underlying infrastructure which would be managed and maintained directly by the cloud service provider. Saas solutions would always ensure the availability and the security of the software. (Microsoft Corporation s.a.)

The benefits of cloud services include cost efficiency and savings, scalability, flexibility, and disaster recoverability. Organizations can reduce expenses spent on hardware and software by transitioning to cloud services offerings. Additionally, cloud services enable rapid scaling of resources to meet fluctuating demands, enhance collaboration through shared resources, and provide robust backup and recovery solutions. By adopting cloud services, organizations can focus on their core competencies while leveraging the advanced capabilities and infrastructure provided by cloud service providers.

2.4 Infrastructure as Code (IaC)

Infrastructure as Code (IaC) is the process by which computer infrastructural resources are provisioned and managed through code as opposed to the legacy system of manual or interactive configurations. This approach allows for the automation and consistency of infrastructure deployment and management, reducing errors and increasing efficiency.

Infrastructure as Code is a key component of DevOps that combines with further practices like Continuous Delivery and Continuous Deployment to incorporate infrastructural changes with application deployment processes. As IaC is codified in configuration files, it can implement version control in the same way as application code to enable collaboration among teams. It integrates infrastructure management with the development process, promoting a more unified approach. This integration between infrastructure management and software developments leads to quicker and more dependable deployments of applications and the underlying infrastructure. (Amazon.com, Inc. s.a; Red Hat, Inc. 2022; Microsoft Corporation 2022.)

2.4.1 Azure Resource Manager

Azure Resource Manager is a component service of Microsoft Azure that is responsible for the deployment and management of resources. It offers a management layer that enables various data

operations and actions like creating, updating, and deleting resources; managing features, like access control, locks, and tags, to secure and organize resources on Microsoft Azure after deployment. As Figure 2 shows, Azure Resource Manager allows for efficiency, consistency and stability across all Azure tools like APIs, CLIs or SDKs, hereby offering a uniform experience. It also ensures secured access to the various Azure resources and services. Figure 3 illustrates how Azure resources and services can be managed via a hierarchical system, where there is a downward inheritance flow where lower levels inherit the management attributes of their corresponding higher levels. (Microsoft Corporation 2024.)

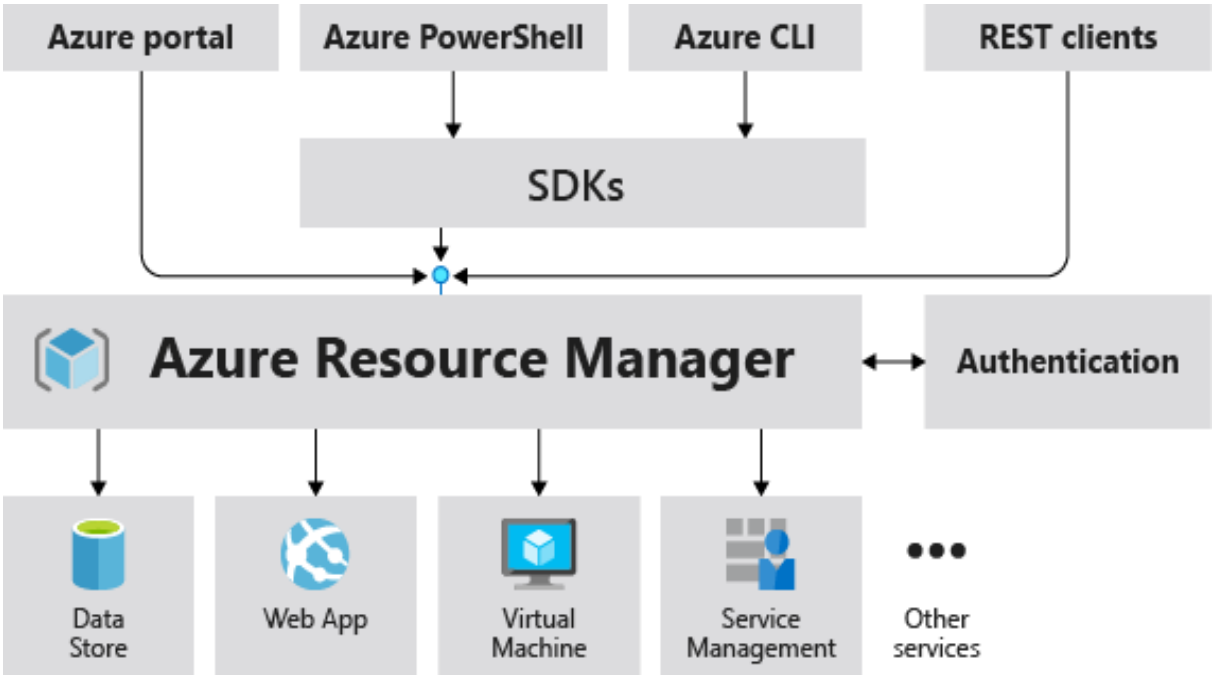

Figure 2. Role of Azure Resource Manager in Handling Azure requests (Microsoft Corporation 2024)

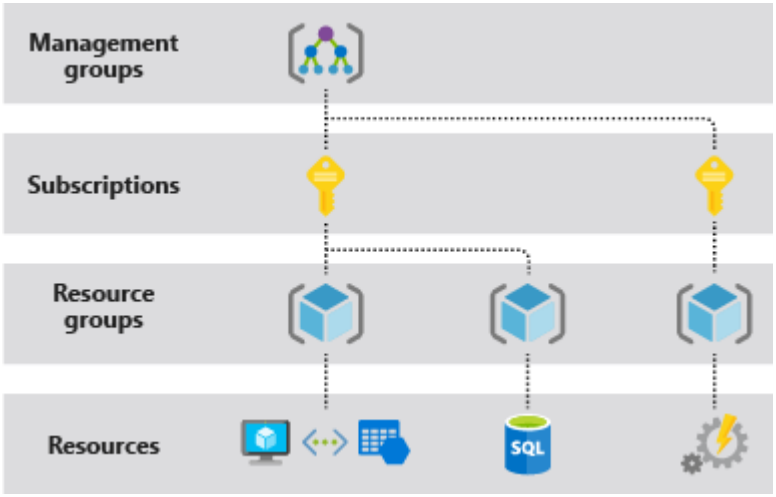

Figure 3. Azure Management Scope Levels (Microsoft Corporation 2024)

2.4.2 Terraform

Terraform is an IaC tool that enables the safe and efficient management of infrastructural resources across various situations and scenarios. Human-readable configuration files, written in the HashiCorp Configuration Language (HCL), can be used to define and configure the resources. Through a

consistent workflow, Terraform manages the entire infrastructure lifecycle, from low-level components like compute, storage, and networking resources to high-level components like Domain Name System (DNS) and Software as a Service (SaaS) offerings. (HashiCorp, Inc. s.a.)

Terraform works by provisioning and managing resources on various cloud platforms and services via their APIs. This is done by Terraform providers which serve as interfaces between Terraform and the available APIs of the cloud platforms and services as noted in Figure 4. Figure 5 shows how Terraform uses a three-stage workflow: writing configurations, planning an execution plan, and applying the changes. This process ensures traceability and efficient provisioning. (HashiCorp, Inc. s.a.)

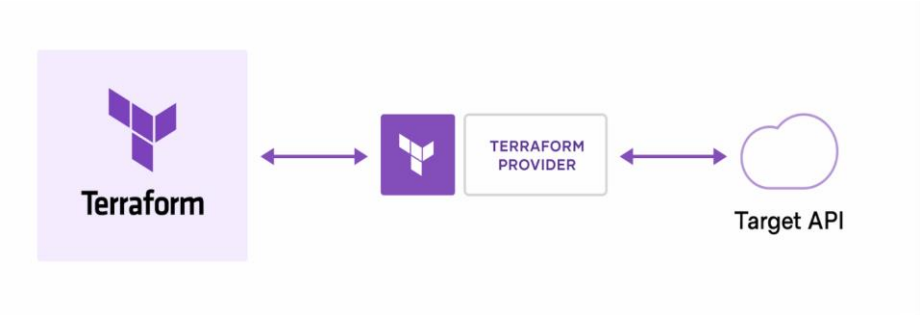

Figure 4. Terraform Overview (HashiCorp, Inc. s.a)

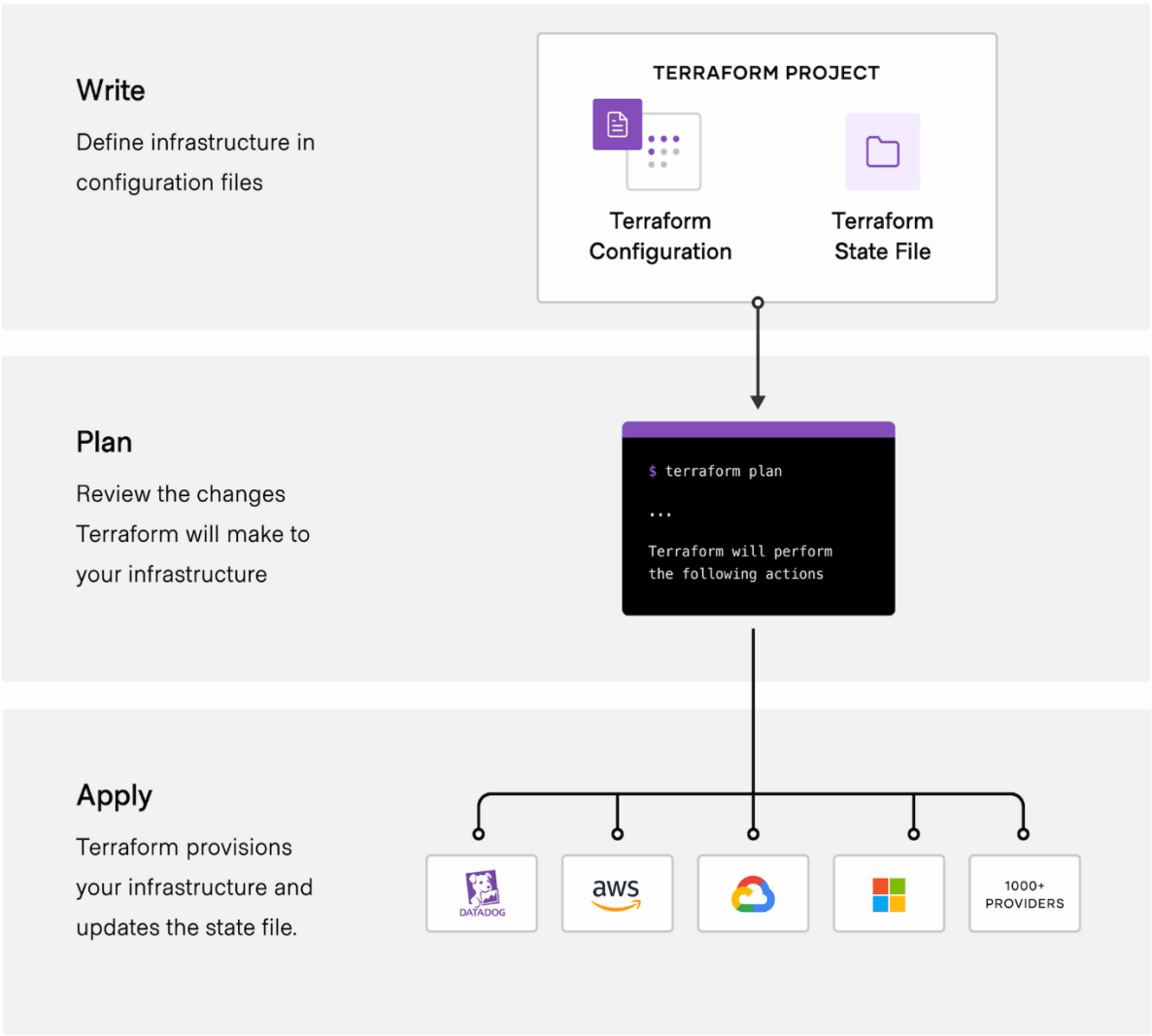

Figure 5. Terraform Workflow (HashiCorp, Inc. s.a)

Terraform offers various advantages like managing any infrastructure with various providers, tracking infrastructure changes through a state file, automating changes with declarative configuration files, standardizing configurations with reusable modules, and facilitating team collaboration through source control management and version control systems. (HashiCorp, Inc. s.a.)

The client organization employs Terraform, using the Azure provider to interface with Azure Resource Manager, as the standard tool for IaC workloads. This integration allows for the automated management and configuration of infrastructure through code. By leveraging a CI/CD pipeline, the deployment process is automated, ensuring consistent and efficient infrastructure changes. This approach aligns with DevOps practices, promoting continuous integration and continuous deployment, while maintaining security and organization of resources within Azure.

3 IMPLEMENTATION

For the practical implementation of this thesis, programming and development work has been done to build and develop a sufficient software projects management system that integrates tightly with DevOps practices to enable the creation and initialization of projects through an automated process and according to the requirements and specifications of the client organization.

As illustrated in Figure 6, the development project is essentially a command-line interface (CLI) tool that is powered by a backend service comprising of application programming interface (API) endpoints that integrates with an Azure Cosmos DB database, a serverless Azure Functions service and API endpoints of some external services like SonarQube and JFrog Artifactory. The implementation has been written in the C# programming language and utilizes version 6 of the ASP.NET Core web application framework. Additionally, there is an incomplete implementation of a web-based graphical user interface (GUI) written in the Typescript programming language using version 17 of the Angular web application framework that would ideally replace the CLI if development on this topic were to continue. The web application is containerized and deployed to Ponsse's Azure Service Fabric platform. Code artifacts are further stored on Ponsse's JFrog Artifactory platform and static program analysis is done through Ponsse's SonarQube server for continuous code quality checks.

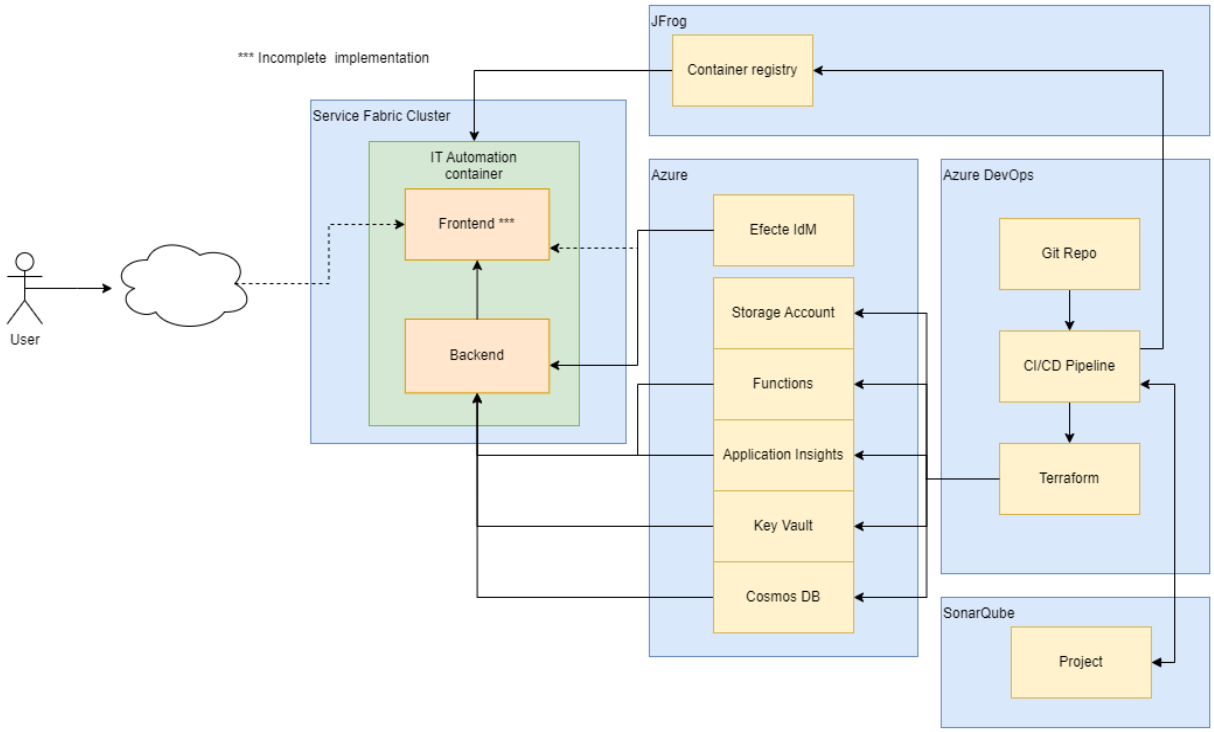

Figure 6. System Overview (Lawal 2024)

The command-line interface (CLI) is intended to serve as a temporary solution for end-user interaction. This approach has been adopted due to the limited knowledgeability and familiarity of the author, with the Angular web application framework and the TypeScript programming language, which are the standard technologies for front-end development within the client organization. Additionally, the scope and time constraints of the thesis did not permit an adequate acquisition of knowledge in these technologies. Consequently, a simple CLI, developed in C#, the same programming language utilized for the backend, was opted for. The front-end interface, intended for end-user interaction,

should ideally be transitioned to, or implemented as a web-based graphical user interface (GUI), as noted in Figure 7, should Ponsse choose to continue pursuing the development of this thesis.

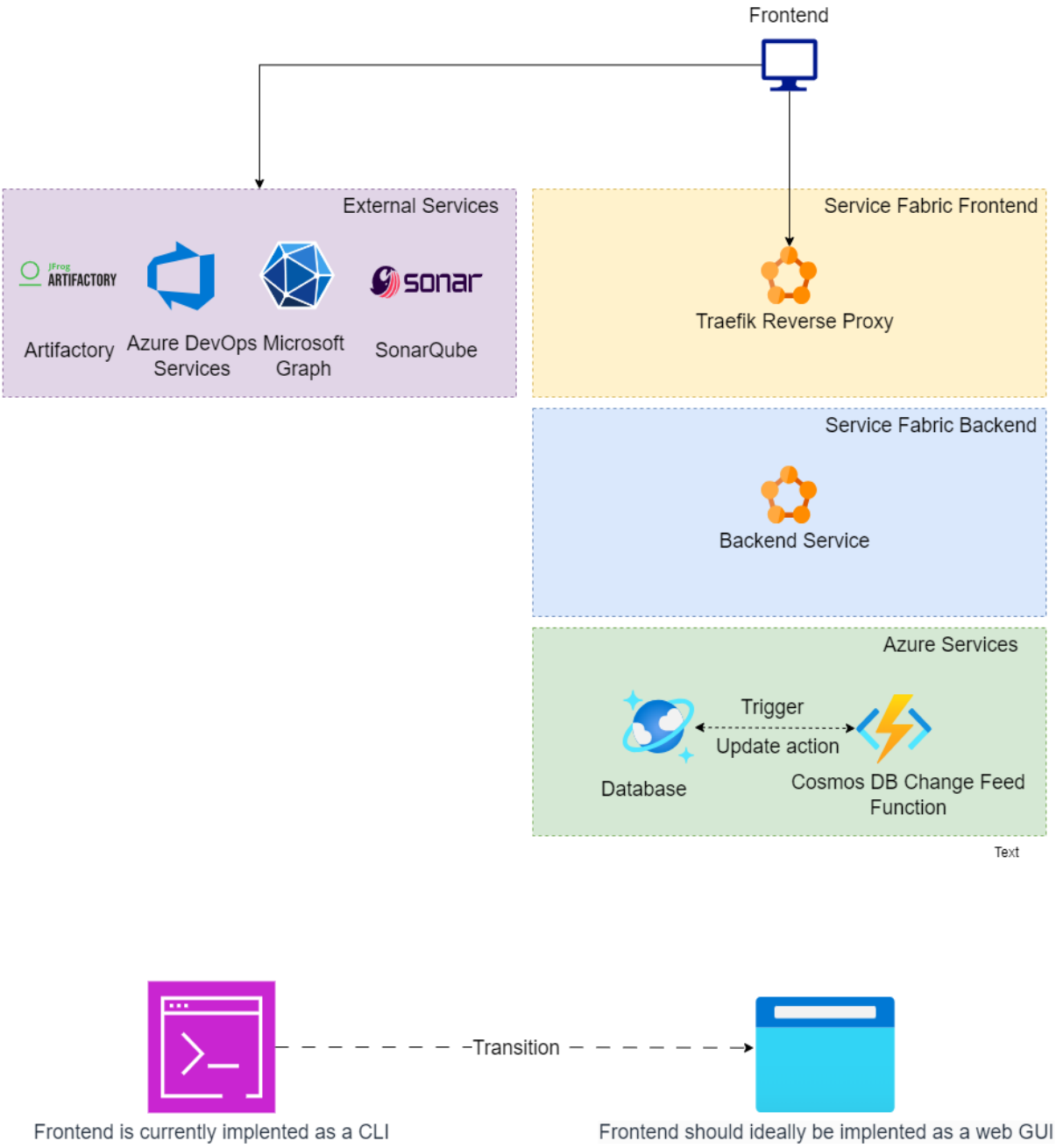

Figure 7. Implementation State (Lawal 2024)

3.1 Git Workflow

The Git workflow employed in the project is gitflow, as specified by the client organization. This workflow includes two permanent or long-lived branches, namely 'main' and 'develop'. The 'main' branch, maintained in a state ready for production, is always deployable, reflecting its critical role. Conversely, the 'develop' branch serves as the integration branch for features before they are released.

Release branches, adhering to the naming convention "release/<version-number>", exist only during the version stabilization phase. These branches facilitate focused preparation for the upcoming release without impacting the ongoing development activities on the 'develop' branch.

Direct git commits to the 'main' and 'develop' branches are not permitted; instead, changes must be introduced through pull requests. This approach is enforced by policies set on the Azure DevOps repositories, which ensure that each pull request is reviewed by at least two individuals. Furthermore, these policies mandate that all comments on the pull requests be resolved, and that the code be validated by pre-merging and building the pull request changes. Additionally, the title of each pull request must begin with a Jira Issue ID, a requirement facilitated by webhooks and an internal Ponsse SCM Jira service. Importantly, these policies also restrict the type of git merge that can be used for the pull requests, ensuring consistency and adherence to the project's development standards.

Trunk-based development is a practice in version control management where developers frequently merge small updates into a central "trunk" or main branch. This technique is widely adopted by DevOps teams and is integral to the DevOps lifecycle, as it simplifies the merging and integration processes. This practice is basically a way of collaborating on a codebase in a single trunk branch that enables developers to avoid merge hell and does not break the build. (Atlassian Corporation s.a; Trunk Based Development s.a.)

While gitflow is currently in place due to its alignment with Ponsse's existing practices, there is consideration for transitioning to trunk-based development in the future. Trunk-based development, recognized for its compatibility with DevOps practices, is already being adopted for some projects within the client organization. This approach could potentially streamline processes and enhance efficiency in software development and deployment cycles.

3.2 CI/CD Pipelines

The project contains two pipelines, one each for the application and the infrastructure, implemented using Azure Pipelines and hosted on Ponsse's Azure DevOps Services organization. The project does not currently include a pipeline for test automation purposes as Ponsse already has plans for the development and integration of test automation to its software development projects. This project has been configured in such a manner that test automation can be included later.

3.2.1 Application Pipeline

Figures 8, 9, and 10 show the process flow for the application pipeline, which has been designed to consider the different branch types across various environments like development, test, staging, and production. Each branch type—feature, develop, release, and main—undergoes specific stages, signifying the transition from development to production.

- For **feature** branches, the pipeline is initiated by a git push against specific files like the application source code, pipeline configuration files, version files and dockerfiles in the application repository. The process workflow triggers a build stage that runs a specific sequence of steps:
    - Build: Compiles the application source code into computer executable code and checks for errors.
    - Unit tests: Ensures that individual components or units of the application source code function as expected.

  - Analyze: Assesses and inspects code quality, styles, and security statically through SonarQube.

  Within Azure Pipelines, the build stage corresponds to the **Development Build** stage. It completes the initial job of checking and testing the code changes in a local development environment on a CI server.

- For the **develop** branch, the pipeline is triggered by a pull request from a feature branch to the develop branch. In this case, the process workflow runs two stages, a build stage that involves similar steps from a pipeline run triggered from a feature branch in addition to other steps:
  - Version: Assigns a unique version number to the application according to the Semantic Versioning scheme.
  - Build: Compiles the application source code into computer executable code and checks for errors.
  - Unit tests: Ensures that individual components or units of the source code function as expected.
  - Analyze: Assesses and inspects code quality, styles, and security statically through SonarQube.
  - Set Manifest Version: Updates the version number in the Service Fabric manifest files.
  - Containerize Application: Packages the application into a container image for deployment.
  - Upload Artifacts: Stores the build artifacts in JFrog Artifactory for use in further stages.
  - Scan Artifacts for Vulnerabilities: Checks for any security vulnerabilities within the artifacts with the JFrog Xray tool.

  that prepares the application for the deployment stage:
  - Download Artifacts: Retrieves the build artifacts from JFrog Artifactory for use in the deployment stage.
  - Update Manifest: Sets the container image details in the Service Fabric manifest files.
  - Deploy: Publishes the containerized application to the test environment on a non-production instance of Service Fabric.

  Within Azure Pipelines, the build stage corresponds to the **Test Build** stage while the deployment stage corresponds to the **Test Deployment** stage. These focus on building and deploying in the test environment.
- For **release** branches, the pipeline is triggered by a pull request from the develop branch to a release branch. Like the develop branch, the process workflow runs the similar stages, with added emphasis on deploying the application to the staging environment on a non-production instance of Service Fabric.

  Within Azure Pipelines, the build stage corresponds to the **Staging Build** stage while the deployment stage corresponds to the **Staging Deployment** stage. These focus on building and deploying in the staging environment.

- For the **main** branch, the pipeline is triggered by a pull request from a release branch to the main branch. As the main branch is a production-ready branch, it requires manual approval check from required persons before deployment can be completed. Like the develop and release branches, the process workflow runs the similar stages, with added emphasis on deploying the application to the production environment on a production instance of Service Fabric. This represents the final point in the software delivery process.
  Within Azure Pipelines, as shown in Figure 9, the build stage corresponds to the **Production Build** stage while the deployment stage corresponds to the **Production Deployment** stage. These focus on building and deploying in the production environment.

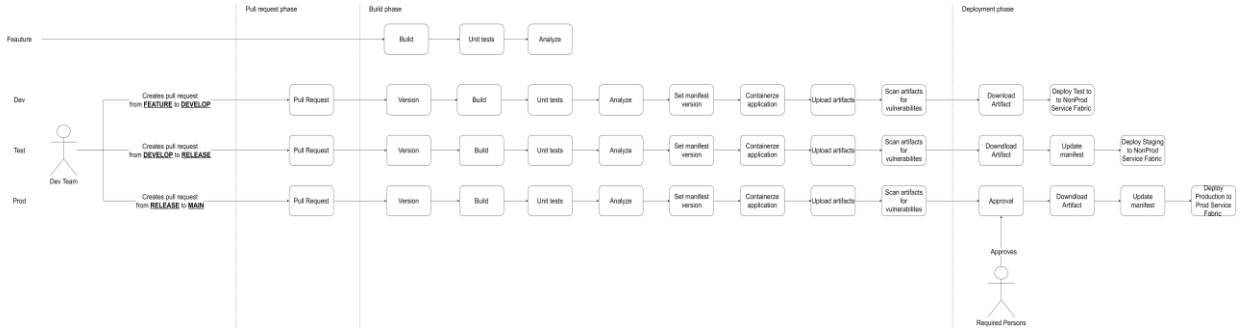

Figure 8. Process Workflow for the Application Pipeline - Gitflow (Lawal 2024)

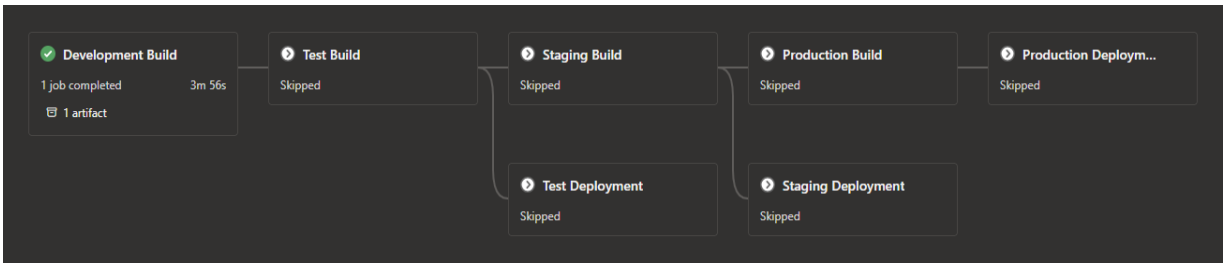

Figure 9. Application Build and Deployment Stages in Azure Pipelines - Gitflow (Lawal 2024)

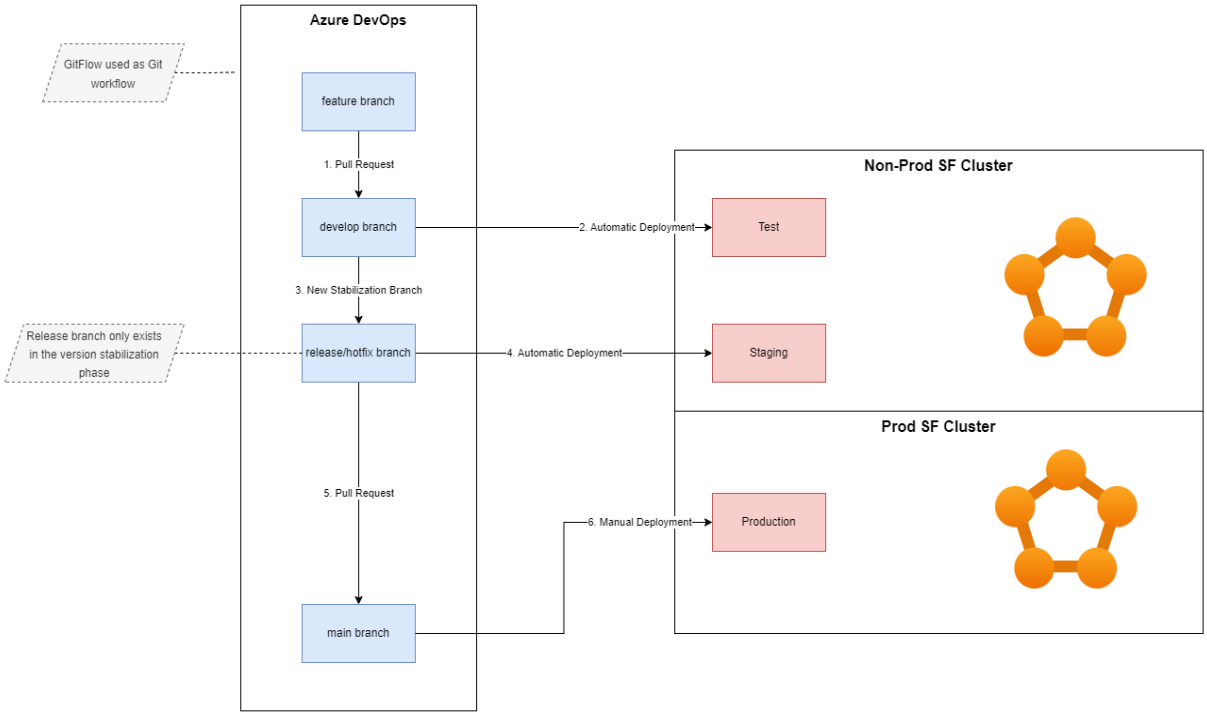

Figure 10. Deployment Overview for the Application - Gitflow (Lawal 2024)

The existing pipeline flow, as utilized in the project for branch-based development utilizing the gitflow git workflow, is comprehensively described above, encompassing a structured approach for deploying applications across different environments from development through production. Transitioning to trunk-based development, as the client organization contemplates this shift, would necessitate a revision of the current workflow to accommodate a streamlined process focusing primarily on a single source of truth for deployment—namely, the trunk or main branch.

Feature branches are used for all development work but are merged back into the **main** (trunk) branch through pull requests frequently. Upon each pull request, the pipeline is triggered automatically, and the process workflow runs a build stage and environment specific deployment stages, shown in Figure 12, which entails similar existing steps as discussed above, with the significant difference being that all deployments are done off a singular build regardless of the environment.

The provided Figures 11, 12, and 13 illustrate these processes within Azure Pipelines, depicting the streamlined flow from code commit through deployment in various environments. This visual representation aids in understanding the operational dynamics under the trunk-based development model, ensuring clarity in the transition process from the existing gitflow approach.

These modifications to the pipeline, proposed in alignment with the transition to trunk-based development, are designed to enhance the agility and efficiency of the development process while maintaining rigorous standards for quality and security.

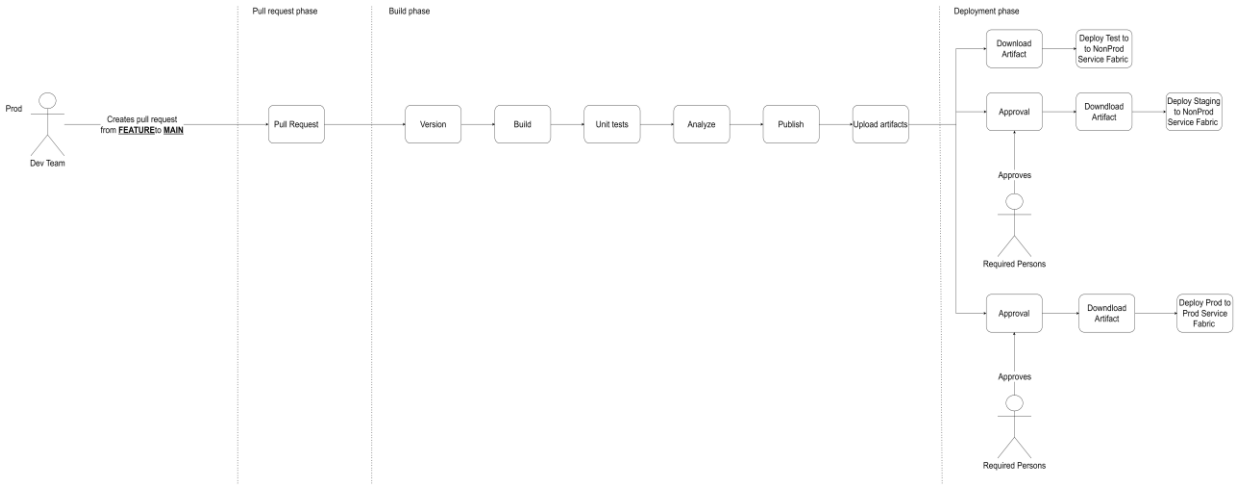

Figure 11. Process Workflow for the Application Pipeline - Trunk-based Development (Lawal 2024)

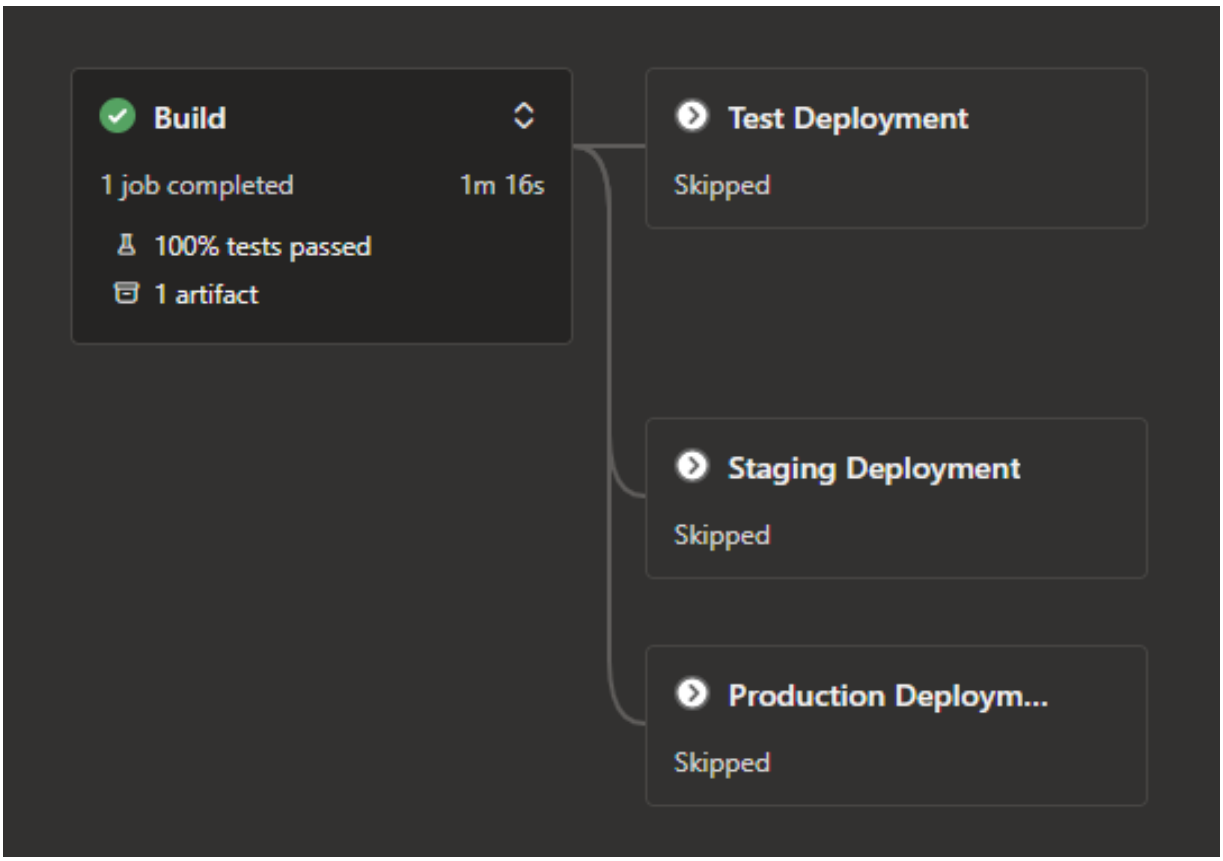

Figure 12. Application Build and Deployment Stages in Azure Pipelines - Trunk-based Development (Lawal 2024)

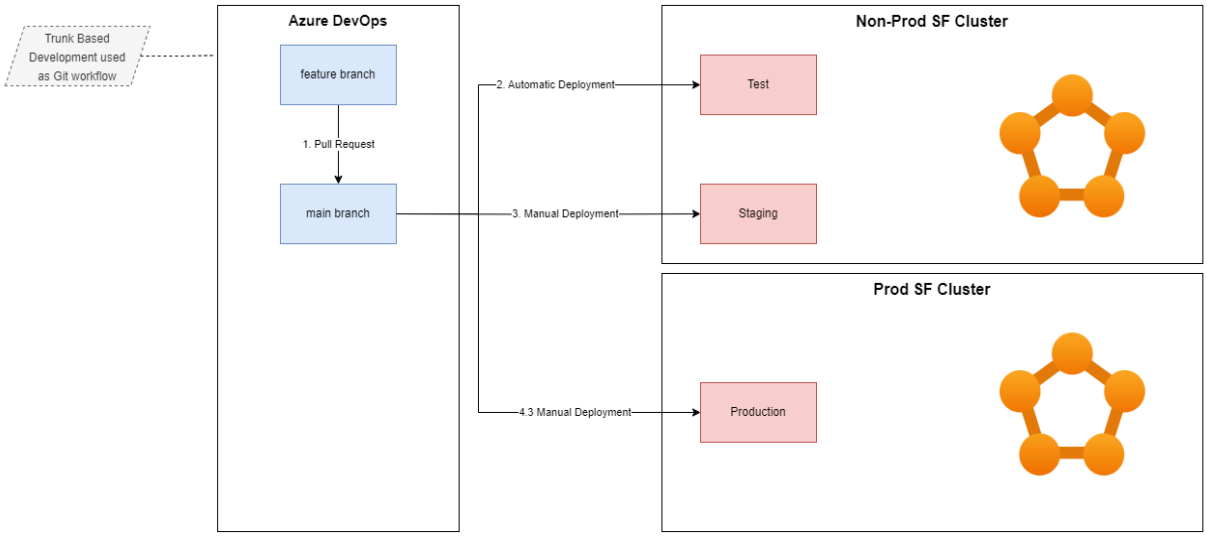

Figure 13. Deployment Overview for the Application - Trunk-based Development (Lawal 2024)

3.2.2 Infrastructure Pipeline

As depicted in Figures 14, 15, and 16, the process flow for the infrastructure pipeline has been designed to deploy infrastructure changes across various environments like test, staging, and production on Microsoft Azure using Terraform with approval and validation mechanisms. The pipeline is tailored to ensure that all deployments, especially in production, are secure, stable, and align with organizational policies and governance. By integrating an approval process and validation steps, the workflow minimizes risks associated with infrastructure changes and enhances the reliability of deployments across all environments.

The process workflow includes the following key components and steps:

- Developer Action:
    - Creates a pull request from any branch to the main branch. This initiates the process for potential changes to be merged into the **main** branch, which is crucial for **staging** and **production** deployments.
- Approval Process:
    - Checks any changes being merged or deployed. This involves a review process by required personnel to ensure all changes meet the necessary standards and requirements for production readiness. This is a requirement for deployments to the production environment.
- Configuration and Initialization:
    - Configure Resource Group and Storage: Utilizes Azure CLI to ensure the necessary infrastructure components like resource groups, storage accounts, blob containers, Microsoft Entra (previously Azure Active Directory (Azure AD)) (Microsoft Corporation 2024) security groups and roles assignments are properly configured or created if they do not exist. This step is crucial for maintaining structured access and storage capabilities within Microsoft Azure.
    - Terraform Init: Initializes the Terraform workspace, configures the backend for storing the Terraform state and prepares the tool for configuration validation, planning and application.
- Validation and Planning:
    - Terraform Validate: Validates the Terraform configuration to ensure all codes are correct and will execute as expected without errors.
    - Terraform Plan: Generates a detailed execution plan that outlines the actions Terraform will take based on the configuration files. This plan is essential for reviewing potential changes before they are applied.
- Conditional Application of Changes:
    - A decision point follows the Terraform Plan step, where it is determined whether the changes should be applied:
        - True: If the plan is approved and conditions are met, the changes are applied and deployed by Terraform to Microsoft Azure.
        - False: If the conditions are not met or the plan is rejected, the process stops, and no changes are applied and deployed by Terraform to Microsoft Azure.

The pipeline is designed to manage deployments across various environments by adjusting the flow based on the branch type:

- For the **test** environment, the pipeline can be run from any branch without the need for approval and validation from any required persons.
- For the **staging** environment, the pipeline must be run from the **main** branch without the need for approval and validation from any required persons.

- For the **production** environment, the pipeline must be run from the **main** branch after approval and validation from any required persons.

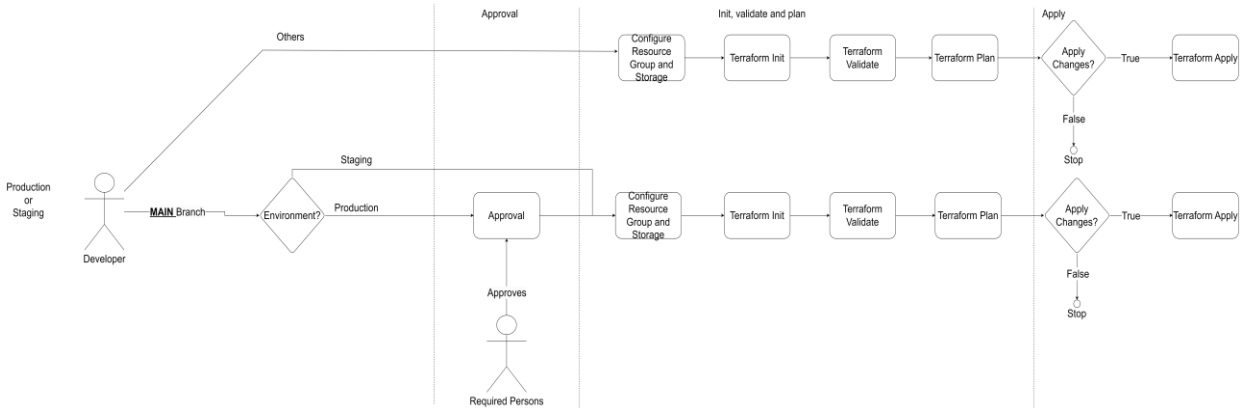

Figure 14. Process Workflow for the Infrastructure Pipeline (Lawal 2024)

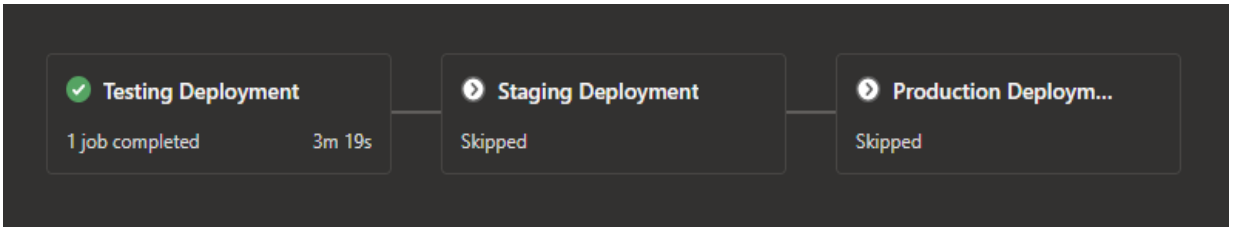

Figure 15. Infrastructure Deployment Stages in Azure Pipelines (Lawal 2024)

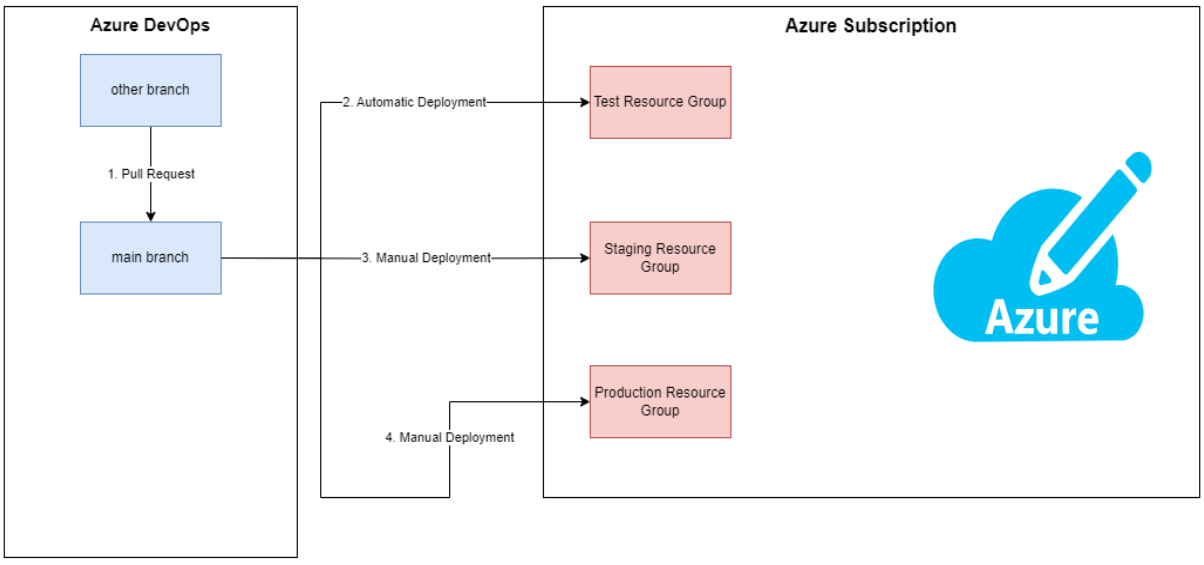

Figure 16. Deployment Overview for the Infrastructure (Lawal 2024)

3.3 Service Fabric

Deployment to Service Fabric is managed by the application pipeline. Two cluster instances of Service Fabric are maintained: Non-Production and Production. The testing and staging environments are deployed to the non-production cluster, whereas the production environment is deployed to the production cluster. Whenever new environments are set up, it is required that secrets and configurations be added to Azure Key Vault and Azure DevOps Library.

Application configurations within Service Fabric are handled through application parameters, while only secrets are stored in Azure Key Vault. Traefik, which is an open-source Edge Router that receives requests on behalf of a system and finds out which components are responsible for handling them (Traefik Labs s.a), runs as a service on both Service Fabric clusters. It serves as a reverse proxy for the application, facilitating the routing of requests to other appropriate services.

Should Ponsse decide to proceed with further development of the project and implement trunk-based development, it is advised that application configuration and secret management be transitioned to a direct management approach using a combination of Azure App Configuration and Azure Key Vault. This change is recommended because the ASP.NET Core framework, utilized in the backend, supports both Azure App Configuration and Azure Key Vault. Such an integration would streamline the configuration process by eliminating the need to link an existing Azure Key Vault to an Azure DevOps variable group, map selective vault secrets to the variable group, and subsequently update the Service Fabric parameters on the pipeline. The proposed combination of Azure App Configuration and Azure Key Vault would enable the application to directly interact with both the key vault and app configuration, configuring itself without the necessity for additional intermediary steps. This adjustment would enhance operational efficiency and reduce complexity in the application's configuration management.

3.4 Microsoft Azure

Microsoft Azure is designated as the chosen cloud provider for the client organization, which maintains a hybrid cloud model. Within this model, specific services and resources are hosted on the organization's private servers, while others are managed on Azure's cloud platform. This approach allows for a flexible and secure distribution of computing resources, optimizing both internal control and the scalability offered by Azure's extensive cloud services. The hybrid model supports a balanced infrastructure that leverages the strengths of both private and public cloud environments, accommodating the diverse needs of the organization's operations.

Infrastructure resources on Microsoft Azure, are to be created utilizing Terraform as Infrastructure as Code (IaC). For this project, an infrastructure repository is maintained within Ponsse’s Azure DevOps Services organization, which contains the Terraform configurations necessary for deployment. A corresponding infrastructure pipeline, as detailed in Figure 17, is used to automatically provision the resources on Microsoft Azure.

Resources are required to be provisioned, and managed specifically for each environment and are organized into resource groups accordingly. This organization method entails that each environment possesses its dedicated resource group containing all environment-specific resources required for the application.

The types of resources to be provisioned include Azure Key Vault, which is utilized for holding application secrets, ensuring secure storage and access to sensitive data. Azure Application Insights is employed for collecting logs, and monitoring the application, providing valuable insights into its performance and health. Azure Blob Storage is designated for holding the Terraform state, facilitating

the management and coordination of the infrastructure's current state. Azure Cosmos DB is provisioned to serve as the primary database for storing application data, providing a flexible and scalable solution for the application's data storage needs. Additionally, Azure Functions are used to execute serverless actions within the backend, enhancing the application's scalability and responsiveness.

These provisions are structured to support efficient and secure management of infrastructure resources, tailored to meet the specific needs of each application environment within the organizational framework.

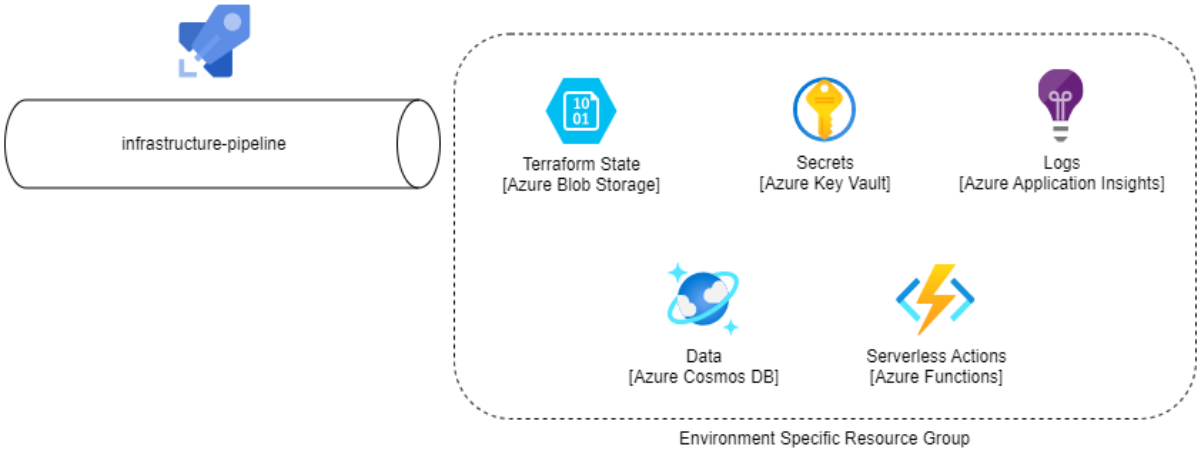

Figure 17. Azure Infrastructure Deployment Architecture (Lawal 2024)

3.5 Front End

The front-end component plays a crucial role in interfacing with the users, providing a command-line interface (CLI) that facilitates the management of digital services platforms and software development projects within the client organization. This CLI is designed to streamline interactions and automate processes that would typically require more complex manual input, thereby enhancing user efficiency and system usability.

The CLI, as detailed in the Figures 18, 19, and 20 provided below, offers a structured and intuitive command set that allows users to manage various aspects of the IT infrastructure:

- Command Overview:
  - The *'ponsse'* command serves as the entry point for interacting with the system. It provides users with options to manage different entities such as platforms and projects through specific subcommands like *'ponsse platform'* and *'ponsse project'*.
  - The *'ponsse platform'* command allows users to list, show, and manage specific digital services platform configurations. This capability is essential for overseeing multiple platforms and ensuring that each is configured correctly and operating as intended.
  - The *'ponsse project'* command suite enables users to create, update, list, and view details of various projects. It supports detailed options to specify project attributes, including the selection of templates, definition of platform and environment details,

and customization of project-specific settings such as naming conventions and security configurations.

The command-line interface (CLI) is structured to provide immediate feedback and help options, making it accessible even to users who are not deeply familiar with command-line operations. Each command is accompanied by a description and a list of available options and subcommands, guiding the user through the necessary steps to achieve the desired outcomes. This interface integrates seamlessly with backend services and databases, such as Azure DevOps and Cosmos DB, allowing for the automated flow of information and actions based on user commands. For example, creating a new project automatically triggers backend processes to set up necessary resources and permissions according to predefined templates and configurations.

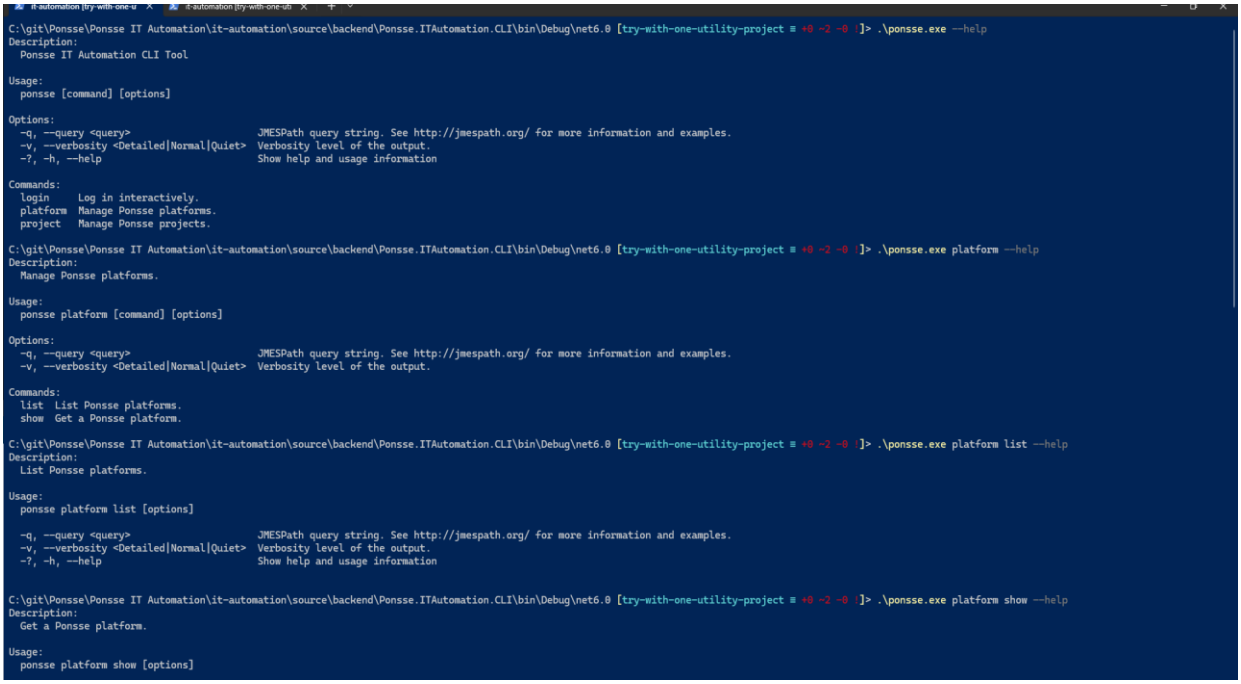

Figure 18. CLI Commands Help Options 1 (Lawal 2024)

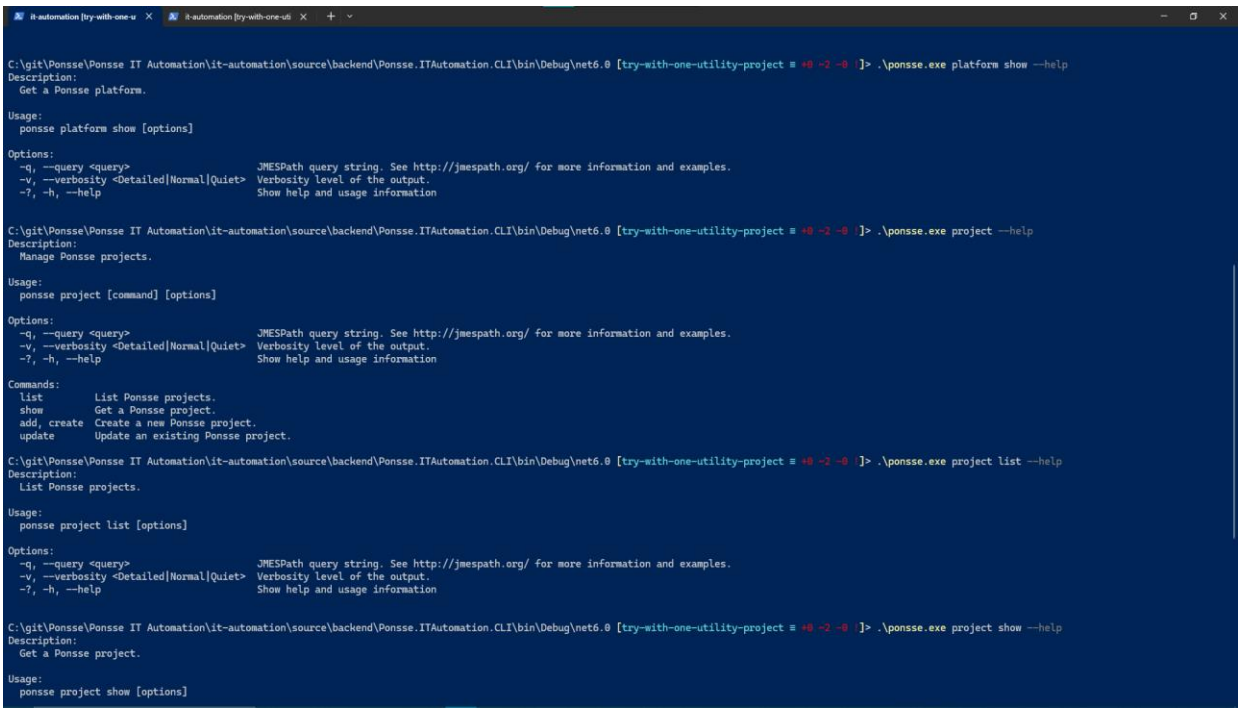

Figure 19. CLI Commands Help Options 2 (Lawal 2024)

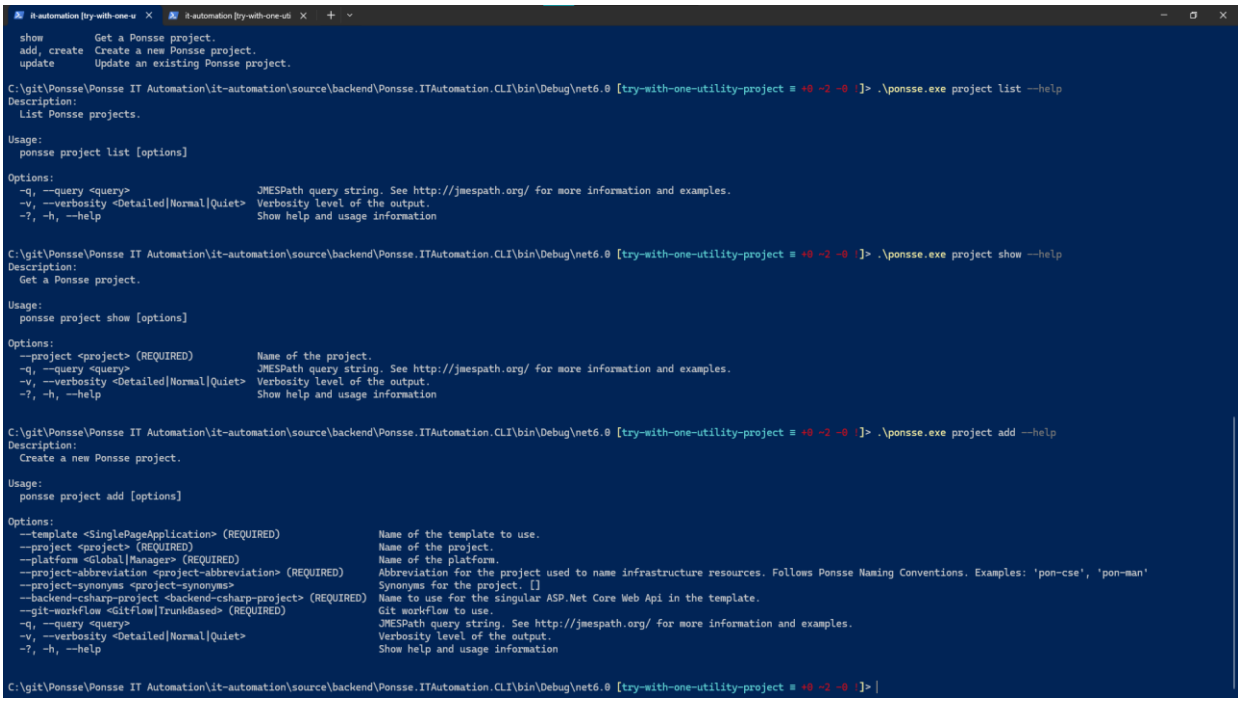

Figure 20. CLI Commands Help Options 3 (Lawal 2024)

While the current CLI effectively supports basic and some advanced functionalities, transitioning to a web-based graphical user interface (GUI) could further enhance user interaction and satisfaction. A web GUI would provide a more visual and interactive experience, potentially lowering the barrier to entry for less technical users and enriching the overall usability of the system.

3.6 Back End

Each of these backend components plays a pivotal role in ensuring the robustness, security, and efficiency of the application infrastructure, forming a cohesive system that supports scalable and secure applications.

3.6.1 REST API

The REST API component serves as a critical interface between the front end and the back-end data storage, specifically connecting to a Cosmos DB instance. This Cosmos DB hosts two containers: "Projects" and "Platforms". The API facilitates various operations on the data stored within these containers. Swagger documentation is generated automatically for the API endpoints, ensuring that they are well-documented and easier to understand for developers.

The Projects container in Cosmos DB holds configuration and information details pertinent to the client organization's software development projects. This includes comprehensive data on project specifications, versions, DevOps configurations, application environments, and access control. Storing data in such a structured way is crucial for tracking project progress and managing the lifecycle of software development within the organization.

The Platforms container stores detailed information about the client organization’s digital services platforms. This encompasses specific details about the projects associated with each platform, effectively cataloging how individual projects integrate with and support various platform functionalities. Additionally, this container maintains data on Microsoft Entra (formerly Azure Active Directory (Azure AD)) (Microsoft Corporation 2024) security groups and on-premises Active Directory (AD) security groups for each project, which are organized into roles specific to each platform. This structured organization aids in access management and ensures that permissions and roles are clearly defined and implemented across projects and platforms.

These containers serve as central repositories for managing extensive data related to the client organization's projects and platforms, facilitating effective data retrieval, manipulation, and management through the provided REST API endpoints. This architecture supports the organization's ability to monitor, update, and control project and platform details dynamically and securely.

This REST API component is implemented to ensure efficient and secure interactions with the Cosmos DB, providing robust support for a variety of database operations essential for managing the application’s data effectively.

3.6.2 Identity

Identity management in the application is crucial for ensuring secure access and operation within the client organization's IT ecosystem. This is managed through an integrated approach involving the organization's Identity Management (IDM) system, which is provided and managed by Efecte.

The application employs OAuth2 Authorization Code Flow, a robust and widely adopted protocol for secure authorization. As illustrated in Figure 21, this method involves redirecting users to an authorization server (managed by Efecte) to log in. Upon successful authentication, the server redirects back to the application with an authorization code that can then be exchanged for an access token. This token is used to make API requests, ensuring that all interactions are authenticated and secure. Alongside standard authentication processes, the application implements role-based authorization. This strategy involves defining roles and permissions associated with those roles within the organization’s IDM. Users are granted access to specific functionalities within the application based on

their assigned roles, which are managed and verified through the IDM system. The application ensures that only authorized users can access sensitive functions and data, significantly reducing the risk of unauthorized access. (Auth0, Inc. s.a.)

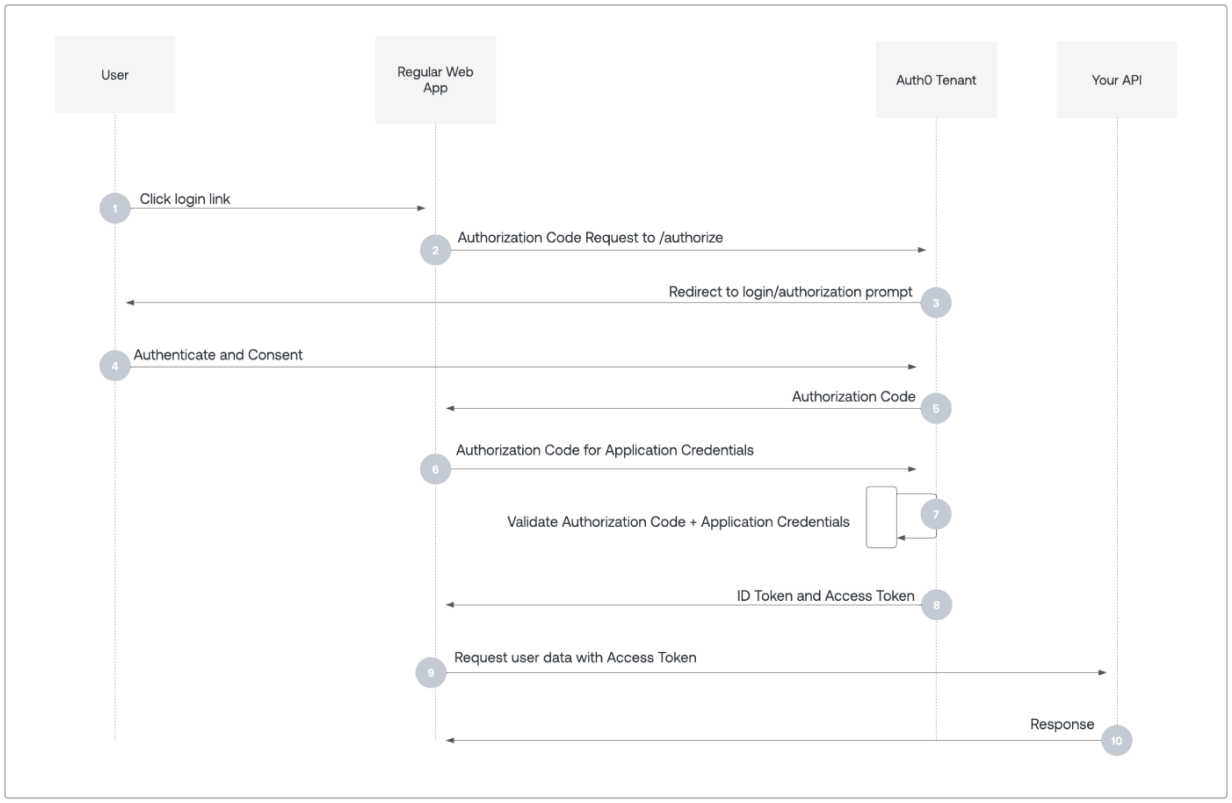

Figure 21. Oauth2 Authorization Code Flow (Auth0, Inc. s.a)

The OAuth2 protocol and the centralized role management provided by Efecte IDM allow for easy scalability. As the organization grows and roles change, the system can adapt without significant overhauls to the application's core authentication mechanisms. Additionally, the application can maintain consistent authentication and authorization strategies across all digital platforms within the organization, simplifying management and enhancing security coherence.

The Efecte IDM system serves as the central point for managing user identities, roles, and permissions. Integration with this system allows the application to leverage a unified identity framework, facilitating seamless and secure user authentication and role validation across the client organization’s various IT services and platforms. This structured approach to identity management not only secures the application but also aligns with best practices for modern application security, ensuring compliance with industry standards and regulations.

3.6.3 Automated Workflow

The automated workflow implemented within the project is designed to streamline the systematic initialization and creation process for software development projects in alignment with the client organization's standardized templates and specifications. This structured workflow is crucial in automating the setup process, ensuring consistency, and enforcing security practices across software development project initialization. Currently, a single template exists for a Single Page Application

(SPA), which adheres to a specific structure and set of practices. This template mirrors the organizational standards across various aspects such as folder structure, which is shown in Figures 22 and 23, pipelines, access control, DevOps practices, Service Fabric deployment, containerization, programming languages, and application frameworks. The same template has been used for the development of this project, and the requirements and specifications explained earlier applies to this project, the SPA template and any other projects that implement this workflow.

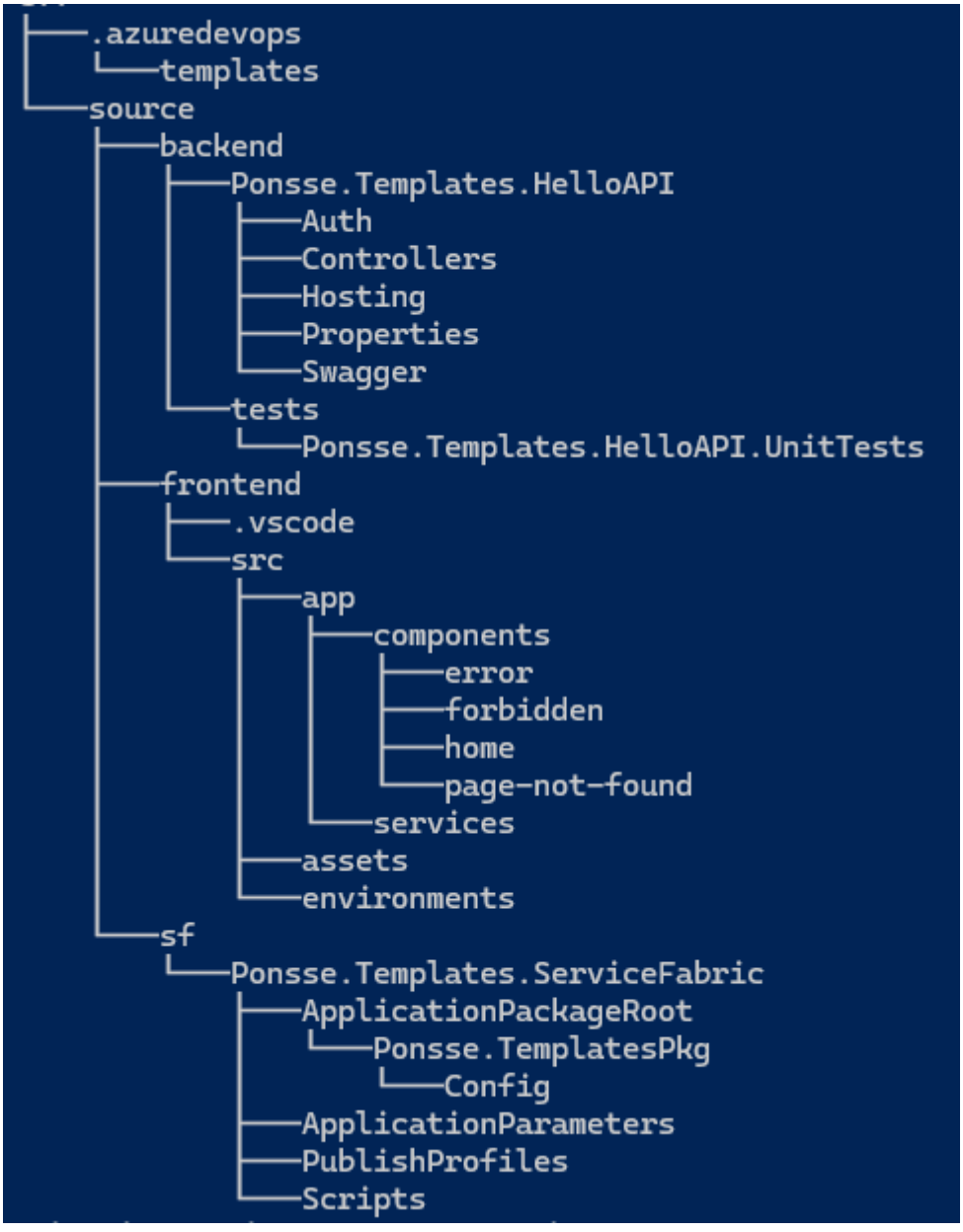

Figure 22. Folder Structure of the Single Page Application (SPA) Template - Application (Lawal 2024)

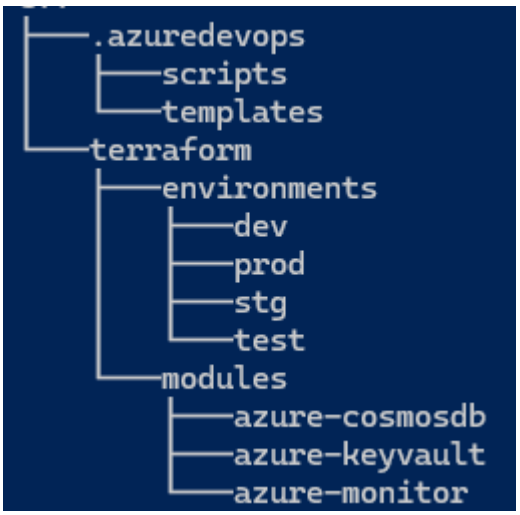

Figure 23. Folder Structure of the Single Page Application (SPA) Template - Infrastructure (Lawal 2024)

As shown in Figure 24, the automated workflow includes the following steps:

- Project Versioning:
  - At the start of the workflow, unique GUIDs necessary for versioning of the project are generated using an internal Ponsse API service. This step ensures each project version is traceable and uniquely identified across any deployment environment, a fundamental requirement for effective version control.
- Microsoft Entra and Active Directory:
  - Essential to the security framework of the project setup, Microsoft Entra (formerly Azure Active Directory (Azure AD)) (Microsoft Corporation 2024) security groups and corresponding on-premises Active Directory (AD) security groups are programmatically created to manage access to critical tools and resources. These resources include JFrog Artifactory, Azure DevOps Services, SonarQube, and Microsoft Azure. The AD security groups are specifically tailored to control access based on roles and responsibilities, ensuring secure and compliant access to project resources.
- SonarQube:
  - Within the workflow, permissions for the SonarQube project are predefined and applied through a permission template configured in the code logic. The SonarQube AD group, created earlier in the process, is utilized to regulate access to the SonarQube project according to Figure 26. Notably, the creation of SonarQube projects is automated and triggered by the initial run of the application pipeline, ensuring seamless integration and minimal manual intervention.
- JFrog Artifactory:
  - The workflow automates the creation of two specialized repositories in Artifactory—one for generic artifacts and another for Docker images. Following their creation, permission targets are set up, and necessary permissions are assigned to manage access through the JFrog Artifactory AD group created earlier in the process, as shown in Figure 26. This automation ensures that repository settings align with security policies and project requirements.

- Azure DevOps:
    - The automated workflow extends to setting up an Azure DevOps Project along with essential components such as repositories and pipelines for application, infrastructure, and test automation. Access control, repository policies, permissions, and security settings are configured automatically, using teams containing the correct Azure AD groups shown in Figure 25 to manage permissions effectively. Integration with an internal Ponsse SCM Jira service via a webhook ensures compliance with a particular project management protocol which checks the inclusion of Jira Issue IDs in pull request titles. Service connections and a common variable group from the Azure DevOps project containing the templates are shared across the newly created Azure DevOps project. This step ensures consistency in environment configurations and access privileges, streamlining further development efforts.
- Project Configuration:
    - At the culmination of the workflow, all relevant project configuration and access control details are compiled and formatted into a JSON document. This document is then stored in Cosmos DB using the REST API component, providing a permanent record of the project's configuration that aids in maintenance, audits, and future scalability.

By embedding this automated workflow within the code logic, the client organization ensures that each project is set up efficiently, securely, and in accordance with established protocols. This approach not only streamlines the project setup process but also reinforces adherence to security and compliance standards, reduces potential for human error, ultimately enhancing the overall integrity and manageability of software development projects.

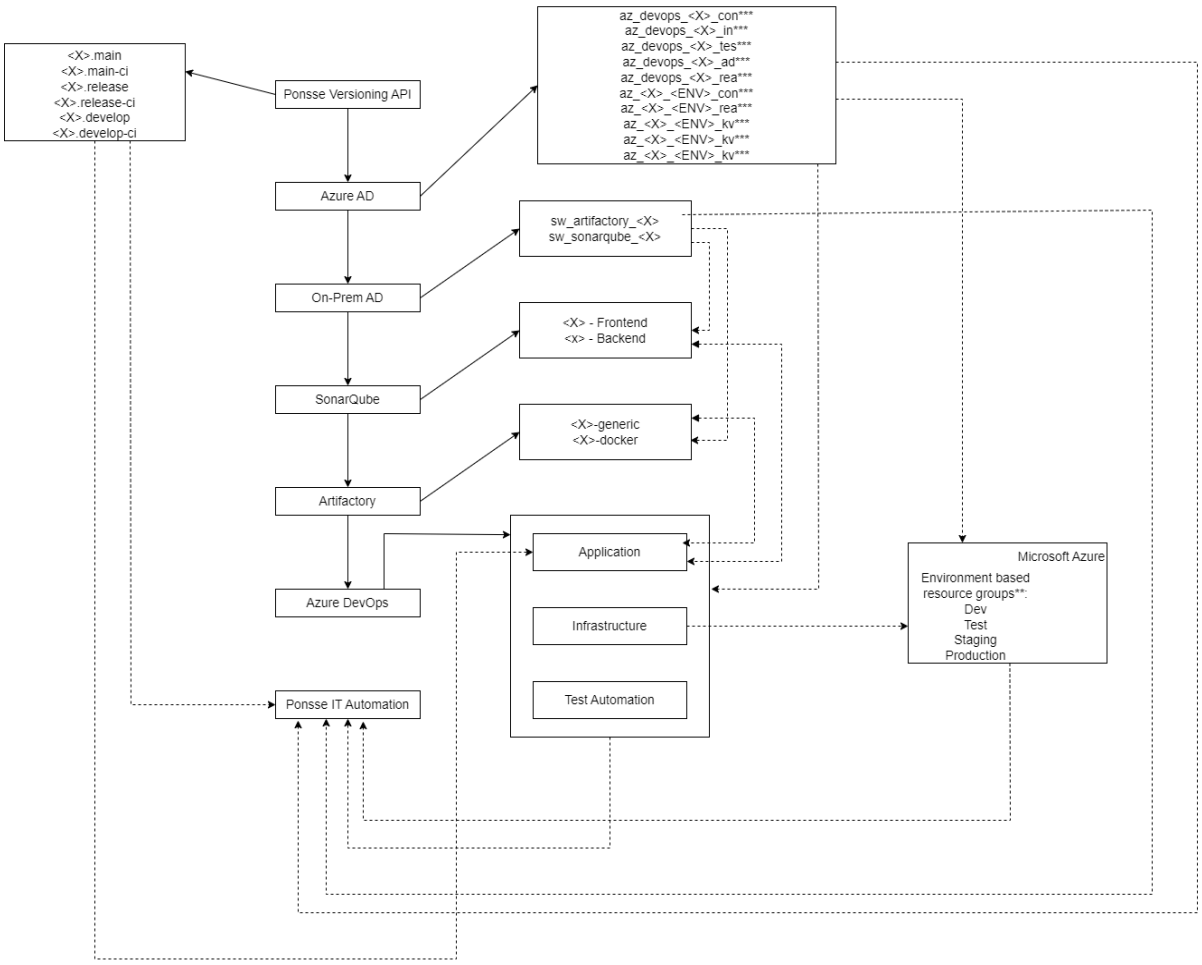

Figure 24. Single Page Application Automated Workflow (Lawal 2024)

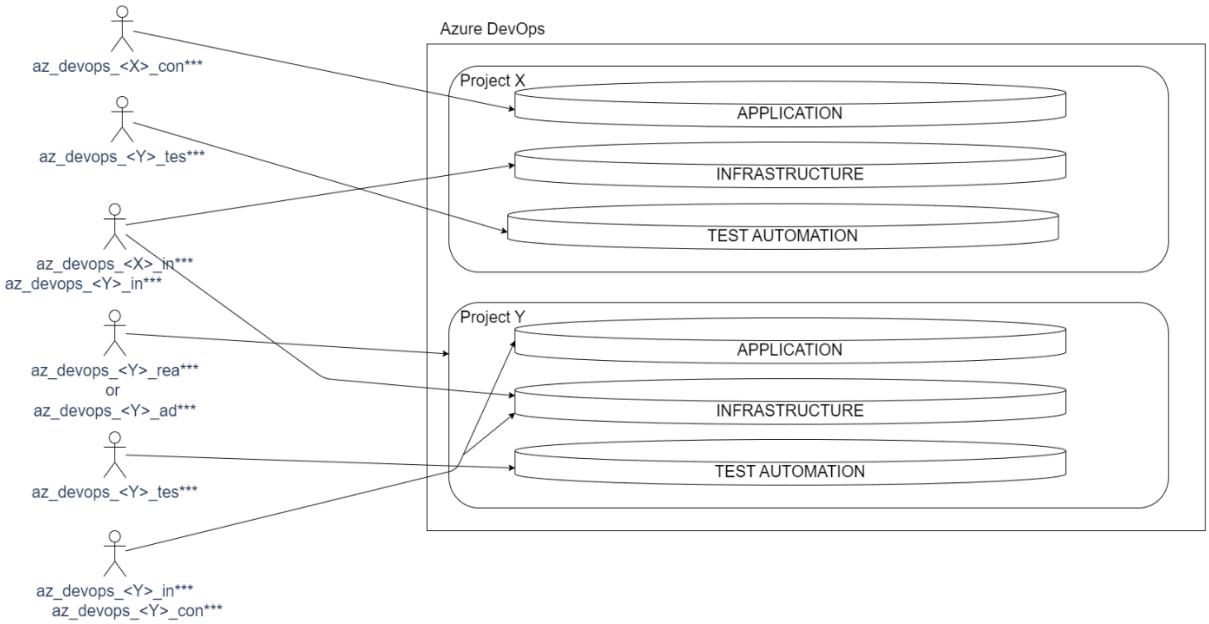

Figure 25. Azure DevOps Access Control (Lawal 2024)

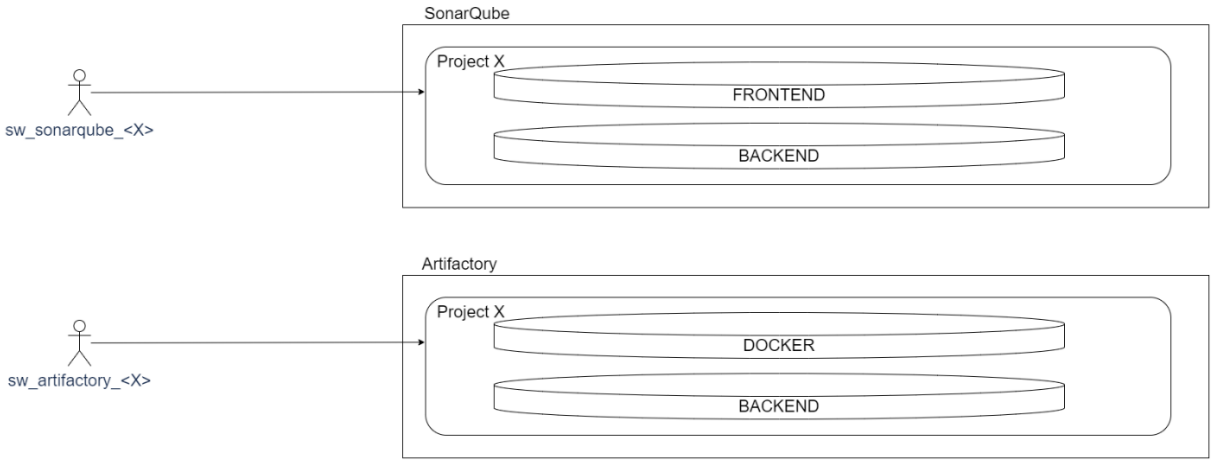

Figure 26. SonarQube and JFrog Artifactory Access Control (Lawal 2024)

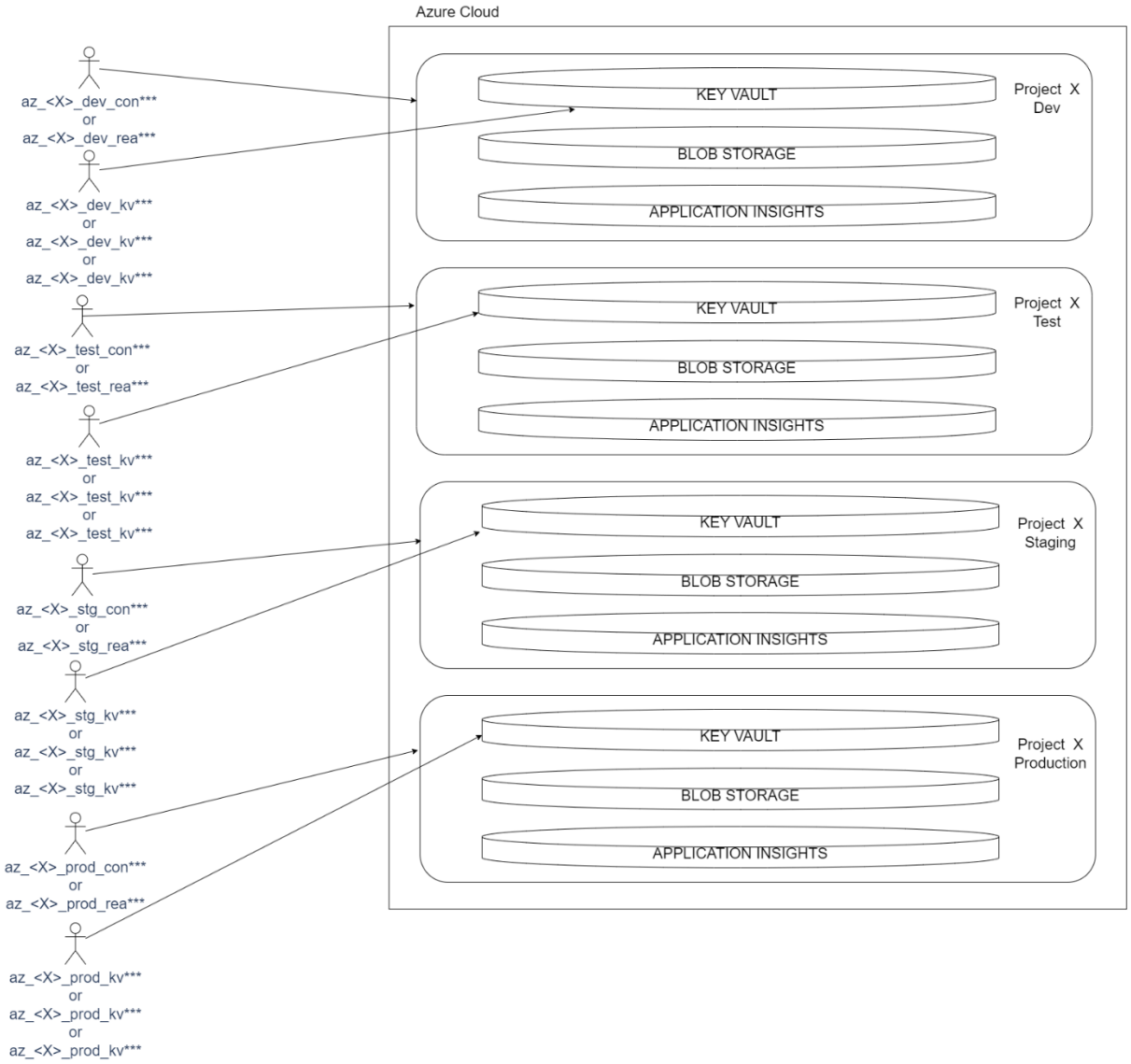

Figure 27. Microsoft Azure Access Control (Lawal 2024)

3.6.4 Cosmos DB Change Feed

The Cosmos DB change feed is a crucial component utilized for monitoring real-time data modifications within the Cosmos DB. This feed plays a pivotal role in integrating with Azure Functions to automate and streamline specific back-end processes.

The Azure Functions service is configured to be triggered by the change feed upon any update action (create or update) in the Projects container. This setup ensures that changes made to project data are immediately captured and processed, facilitating a reactive approach to data management. Once triggered, the Azure Functions service performs specific tasks designed to enhance data synchronization and integrity across different containers within Cosmos DB.

The function extracts essential details from the updated project entries in the Projects container. This includes basic project information such as project name, scope, and relevant dates, along with access control data, such as Microsoft Entra (formerly Azure Active Directory (Azure AD)) (Microsoft Corporation 2024) security groups and on-premises Active Directory (AD) security groups associated with the project. After extraction, this information is then integrated into the corresponding document within the Platforms container. This document pertains to the specific platform associated with the project, ensuring that each platform's document is continually updated with the latest project-related data and access control configurations.

This automated process not only reduces manual data handling errors but also ensures that all related components within the digital service platforms are consistently updated with the latest project details and security configurations. By leveraging the Cosmos DB change feed and Azure Functions in conjunction, the system maintains high data accuracy and integrity, enhancing overall operational efficiency and security compliance. The integration of Cosmos DB change feed with Azure Functions exemplifies a dynamic and automated approach to database management, optimizing the flow of information and ensuring that the system responds swiftly to changes in project data. This setup is vital for maintaining a robust and responsive IT infrastructure within the client organization.

3.6.5 Logging

For effective monitoring and logging, the application leverages Azure Application Insights, a powerful and versatile tool designed to provide comprehensive insights into application performance and user behaviors. The application is integrated with Azure Application Insights to capture a wide range of telemetry data, including requests, responses, exceptions, dependencies, and performance metrics. This integration is critical for gathering detailed insights into the application's operational dynamics in real-time.

Continuous logging and monitoring are essential for maintaining optimal application performance. By analyzing the telemetry data collected by Azure Application Insights, the application's performance can be constantly evaluated and improved. This data helps in identifying bottlenecks, unusual behaviors, or errors that may affect user experience or system stability. Azure Application Insights also contributes to the security and compliance aspects of the application. By logging all interactions and changes, it provides an audit trail that can be used for forensic analysis in case of security incidents. Through the strategic use of Azure Application Insights, the application achieves a high level of operational excellence, ensuring robust performance monitoring and effective logging practices that align with the overall goals of maintaining a stable, secure, and responsive application environment.

3.6.6 Secrets

For secure handling of application secrets, the application utilizes Azure Key Vault, a cloud service specifically designed to safeguard cryptographic keys, certificates, and other secrets used in cloud applications and services. Azure Key Vault is integrated into the application's architecture to manage and protect secrets like API keys, connection strings, passwords, and other sensitive data. The use of Azure Key Vault helps in isolating these secrets from the application's codebase, enhancing security by centralizing the management of these confidential elements. By centralizing the management of secrets, Azure Key Vault plays a crucial role in the security infrastructure of the application. It simplifies the management of secrets and significantly lowers the risk of unauthorized access and exposure.

3.6.7 Settings

For the management of application configurations, Azure DevOps variable groups are utilized in conjunction with Azure Key Vaults, creating a secure and dynamic configuration management system. Azure DevOps variable groups are employed to store and manage settings that are used across multiple pipelines and stages within the CI/CD process. These variable groups centralize the storage of settings, making them accessible and maintainable across various deployment environments. The variable groups in Azure DevOps are linked to various Azure Key Vaults, ensuring that all sensitive configurations stored as secrets in the Azure Key Vaults are seamlessly integrated into the deployment pipelines. This linkage enables secure fetching of secrets directly into the deployment processes without exposing them in the pipeline logs or scripts.

During pipeline executions, these variables are injected into Service Fabric parameter files. These parameter files are crucial as they contain environment-specific configurations that dictate how the application behaves in different environments. By parameterizing settings, the application maintains flexibility and scalability without hard coding environment-specific details. Once passed into the Service Fabric parameter files, the configurations are used to set up and adjust the application configurations within Service Fabric. This method ensures that configurations are dynamically applied based on the current deployment environment, enhancing the application's adaptability to changes and different operational contexts.

Centralized management of settings through Azure DevOps and Azure Key Vault simplifies the operational oversight of the application and upholds stringent security standards. It reduces complexities associated with managing configurations across multiple environments and stages, ensuring consistency and reliability in the application's deployment and runtime behavior. Additionally, prevents unauthorized access and manipulation of critical configuration settings, ensuring that the application complies with security best practices.

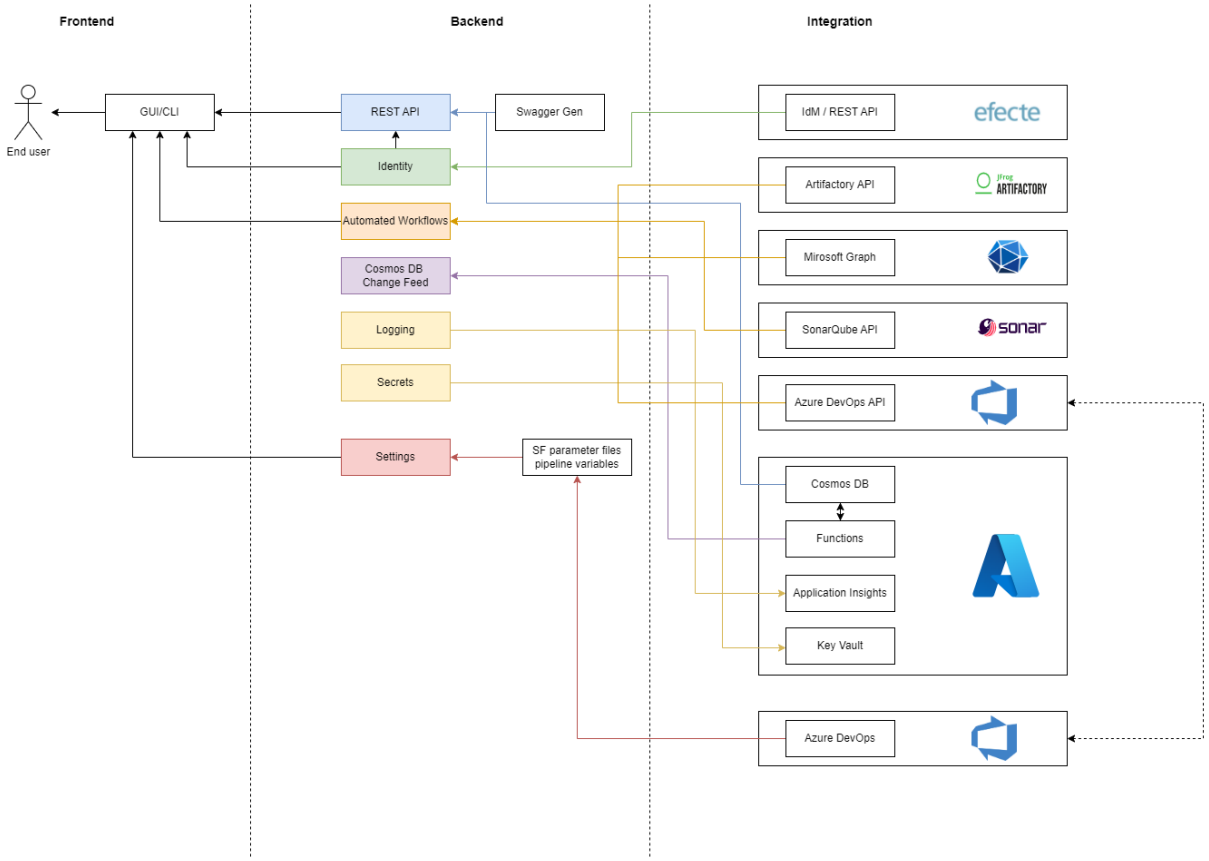

Figure 28. Web Components (Lawal 2024)

4 CONCLUSION

Throughout this documentation, various aspects of managing and automating software development projects within the client organization's infrastructure have been discussed and delved into. The utilization of advanced tools, services and practices like DevOps, CI/CD, Microsoft Azure, automated workflows amongst others, highlights a comprehensive approach to enhancing efficiency, security, and compliance in software development.

The strategies and tools discussed are integral to fostering a controlled, secure, and efficient software development environment. By automating key processes, the client organization not only enhances its operational efficiency but also strengthens its security posture and compliance with regulatory requirements. The detailed exploration of these components provides a roadmap for maintaining and scaling Ponsse's IT infrastructure in a way that meets specified standards and standardized specifications, supports current needs and anticipates future growth and technological advancements.

This thesis successfully implemented a robust software development projects management framework integrating advanced technological solutions mentioned above. These tools and services were effectively utilized to enhance the efficiency and security of the software development lifecycle. Key achievements include the successful deployment of the REST API and Cosmos DB Change Feed, which ensured effective data management and real-time data synchronization. The project also achieved effective integration with Azure services, enhancing operational efficiency and project management capabilities. Despite the success, the implementation of automated workflows for project setup was partially hindered by external constraints related to access control testing and identity management limitations.

The primary goals of the thesis were to develop a framework that could streamline the software development process, enhance security protocols, and integrate modern cloud-based technologies to improve project management and deployment strategies. These goals were largely attained through streamlined project setups using automated workflows that reduced manual processes and potential errors, enhanced security measures with the successful integration of Azure Key Vault and strict access controls using Microsoft Entra (formerly Azure Active Directory) (Microsoft Corporation 2024) and on-premises Active Directory, and improved project management using Azure DevOps, facilitating better tracking, collaboration, and CI/CD practices.

This thesis lays the groundwork for future research in automated project setups and advanced identity management solutions in cloud-based environments. The challenges identified offer fertile ground for exploring alternative Identity Management (IDM) solutions and expanding the capabilities of current systems to handle complex security and operational demands. The reliability of the research conducted is supported by the systematic use of established technologies and methodologies. However, the limitations encountered with Microsoft Entra (formerly Azure Active Directory) and on-premises Active Directory testing due to the client's policy constraints may affect the generalizability of the findings. Further validation in more variable settings would enhance the reliability of the results.

Overall, the thesis successfully met its objectives by implementing the basis of a software development projects management system that leverages modern technological solutions like the cloud and DevOps to streamline operations. While challenges were encountered, particularly in the integration of comprehensive access control testing, the project provided valuable insights into the practical application of cloud technology and DevOps practices in software development. Future enhancements and research are recommended to address the gaps identified, particularly in improving the flexibility and robustness of identity management practices.

5 DISCUSSION

The REST API component was implemented successfully, functioning exactly as intended by facilitating robust interactions with the Cosmos DB. It efficiently handled CRUD operations on the "Projects" and "Platforms" containers, demonstrating the application's capability to manage data effectively. The integration of the Cosmos DB Change Feed with Azure Functions also proved to be highly effective. These functions performed seamlessly, updating corresponding documents in the "Platforms" container whenever changes occurred in the "Projects" container, thus ensuring data consistency and real-time synchronization.

While the integrations of the automated workflow with various services like Microsoft Azure, JFrog Artifactory, SonarQube, and Azure DevOps were successfully implemented and functioned well, challenges were encountered in fully testing and implementing the automated workflow components related to Microsoft Entra (formerly Azure Active Directory) (Microsoft Corporation 2024) and on-premises Active Directory. The client organization's policy of maintaining these directories exclusively in production environments restricted the ability to conduct comprehensive tests in a staging or development setting.

The thesis project also faced significant hurdles in fully implementing authentication and authorization mechanisms due to limitations in the Identity Management (IDM) provided by Efecte. The lack of necessary API endpoints or tools to automate the creation of roles and manage services for OAuth2, as well as handling Microsoft Entra (formerly Azure Active Directory) (Microsoft Corporation 2024) and on-premises Active Directory integrations, hindered the complete realization of this crucial component.

Throughout the thesis process, a significant enhancement in skills related to cloud services management, DevOps practices, and security protocols was noted. The hands-on experience with Azure’s various components and the integration of security measures like OAuth2 for identity management contributed to a deeper understanding of secure software development practices.

The transition to trunk-based development is advocated based on the significant advantages it offers in terms of DevOps and CI/CD efficiencies. This development approach is characterized by developers merging their changes back to the main code trunk branch frequently, minimizing the divergence of branches and reducing the complexities associated with merging long-lived branches. It simplifies the logic behind the CI/CD pipelines for a much more straightforward deployment strategy. It also enables faster iterations and updates which in turn increases and enhances the adaptability of the application. Trunk-based development promotes more frequent updates and integrations, which can lead to faster detection and resolution of conflicts, streamlined testing processes, and reduced time to deployment. The potential for severe integration challenges is significantly decreased, leading to smoother continuous delivery cycles.

The current interaction with users through a command-line interface (CLI) has been functional but does not tap into the potential benefits of a more intuitive and accessible interface. Transitioning to a web-based GUI is recommended to enhance user interaction and satisfaction. A web-based GUI can be more easily scaled and updated with new features and functionalities compared to a CLI,

accommodating more complex interactions and integrations with other web services. As this thesis aims to develop the basis of a software development project management system which is targeted at users like project managers, support specialists etc. who may not be familiar with command-line operations, a web GUI would significantly enhance the user experience. The web GUI would also typically allow for faster execution of tasks with less room for error, as it can guide users through processes with visual cues and structured navigation.

It is advised that the project consider these enhancements and address the noted challenges in its future development strategy to not only keep up with industry best practices but also to adhere to relevant specifications and standards in a rapidly evolving technological landscape. These strategic updates would likely result in greater efficiency, improved user experience, and a more robust solution that is well-adapted to meet the demands of the client organization. These strategic shifts are expected to contribute positively to the project's overall success and sustainability.

REFERENCES

Artificial intelligence has been used in the work as follows:

ChatGPT 2024. OpenAI. GPT-4o. Accessed for language check, June 2024. https://chat.openai.com

Akamai Technologies, Inc. s.a. What Is the Cloud? Internet publication. An Akamai website. https://www.akamai.com/glossary/what-is-the-cloud. Accessed 6.6.2024

Amazon.com, Inc. s.a. What is Infrastructure as Code? Internet publication. Amazon Web Services (AWS). https://aws.amazon.com/what-is/iac/. Accessed 1.5.2024

Atlassian Corporation s.a. Trunk-based development. Internet publication. An Atlassian website. https://www.atlassian.com/continuous-delivery/continuous-integration/trunk-based-development. Accessed 5.6.2024

Auth0, Inc. s.a. Authorization Code Flow. Internet publication. An Auth0 by Okta documentation website. https://auth0.com/docs/get-started/authentication-and-authorization-flow/authorization-code-flow. Accessed 5.6.2024

BrowserStack 2023. What is CI/CD? (Differences, Benefits, Tools, Fundamentals). Internet publication. A BrowserStack website. Updated 29.5.2023. https://www.browserstack.com/guide/what-is-ci-cd. Accessed 6.6.2024

GitLab Inc. s.a. What is CI/CD? Internet publication. A GitLab website. https://about.gitlab.com/topics/ci-cd/#what-is-continuous-integration-ci. Accessed 2.6.2024

HashiCorp, Inc. s.a. What is Terraform? Internet publication. A HashiCorp Developer website. https://developer.hashicorp.com/terraform/intro. Accessed 2.6.2024

International Business Machines Corporation s.a. What are Iaas, Paas and Saas? Internet publication. An IBM website. https://www.ibm.com/topics/iaas-paas-saas. Accessed 6.6.2024

JetBrains s.r.o. s.a. What are the benefits of CI/CD? Internet publication. A JetBrains TeamCity website. https://www.jetbrains.com/teamcity/ci-cd-guide/benefits-of-ci-cd/. Accessed 6.6.2024

Microsoft Corporation s.a. What is IaaS? Internet publication. Microsoft Azure. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-iaas. Accessed 2.6.2024

Microsoft Corporation s.a. What is PaaS? Internet publication. Microsoft Azure. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-paas. Accessed 2.6.2024

Microsoft Corporation s.a. What is SaaS? Internet publication. Microsoft Azure. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas. Accessed 2.6.2024

Microsoft Corporation 2022. What is infrastructure as code (IaC)? Internet publication. Microsoft Learn. Updated 28.11.2022. https://learn.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code. Accessed 1.5.2024

Microsoft Corporation 2024. New name for Azure Active Directory. Internet publication. Microsoft Learn. Updated 5.3.2024. https://learn.microsoft.com/en-us/entra/fundamentals/new-name. Accessed 5.6.2024

Microsoft Corporation 2024. What is Azure Resource Manager? Internet publication. Microsoft Learn. Updated 31.5.2024. https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/overview. Accessed 2.6.2024

Postman s.a. What is an API? Internet publication. A Postman website. https://www.postman.com/what-is-an-api/. Accessed 2.6.2024

Red Hat, Inc. 2022. What is Infrastructure as Code (IaC)? Internet publication. A Red Hat website. Updated 11.5.2022. https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac. Accessed 1.5.2024

Red Hat, Inc. 2023. What is CI/CD? Internet publication. A Red Hat website. Updated 12.12.2023. https://www.redhat.com/en/topics/devops/what-is-ci-cd. Accessed 2.6.2024

Traefik Labs s.a. Welcome. Internet publication. A Traefik Proxy documentation website. https://doc.traefik.io/traefik/. Accessed 5.6.2024

Trunk Based Development s.a. Introduction. Internet publication. https://trunkbaseddevelopment.com/. Accessed 5.6.2024