Thanh Tran Viet Thien

Analysing and Designing CI/CD pipelines in an Enterprise

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

14th May 2024

Abstract

Author:	Thanh Tran Viet Thien
Title:	Analysing and Designing CI/CD pipeline in an Enterprise
Number of Pages:	41 pages
Date:	14th May 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Smart IoT Systems
Supervisors:	Matti Peltoniemi, Senior Lecturer
	Nhut Ha Minh, Product Owner

This thesis delves into the realm of CI/CD pipelines within enterprise settings, focusing specifically on their significance and implementation within a telecommunication organization. The primary objective is to analyze the organization's application development and operational processes, focusing on the implementation of CI/CD pipelines. Through this analysis, inefficiencies and challenges are identified, and practical solutions are proposed to address them. The solution proposal aims to improve the time deployment in infrastructure provisioning in the operation process. It offers servers classifying into two distinct categories: stateful servers and stateless servers. The stateful servers will be implemented as Virtual Network Functions (VNF), while the stateless servers will be deployed as Cloud-native Network Functions (CNF). Furthermore, Telco Cloud Automation serves as the management and orchestration toolset, streamlining application and service deployment and operation.

The scope includes exploring CI/CD pipelines, understanding the company's landscape, analyzing development and operational processes, and proposing implementation strategies. This research aims to streamline operations, minimize manual intervention, and enhance efficiency in software development and deployment within the organization.

Keywords: CI/CD, Stateful, Stateless, containerization, CaaS, Github Actions, Telco Cloud Automation, VNF, CNF Kubernetes

---

Acknowledgement

I would like to express my sincere appreciation to Telia Company, especially the Network Application Department, for granting me the opportunity to undertake this thesis. It has been an invaluable experience that has significantly contributed to my academic and professional growth.

I am extremely grateful to my managers Mr. Harri Hiltunen, Mr. Nhut Ha Minh, and Mr. Jukka Karjaluoto, as well as all the experts at the company, for their unwavering support and guidance throughout this thesis journey. Your collective expertise, encouragement, and patience have played a crucial role in shaping the outcome of this work.

Special appreciation goes to my mentor, Mr. Matti Peltoniemi, your dedication, and expertise have been invaluable throughout this endeavour. Your advice and suggestions have had a profound impact on the completion of this thesis.

Furthermore, I would like to express my gratitude to all those who have supported me in various ways, whether through encouragement, feedback, or understanding. Your contributions have been integral to the successful completion of this thesis.

Lastly, I would like to extend my deepest thanks to my family, my lovely partner and friends for their unwavering encouragement and belief in my abilities. Your love and support have been my anchor during this academic pursuit. I am truly grateful to every one of you for being a part of this journey.

Sincerely

Thanh Tran

Contents

List of Abbreviations

CaaS: Containerized as a Service

CI: Continuous Integration

CD: Continuous Delivery, Continuous Deployment

CM: Configuration Management

CNF: Cloud-native Network Function

CNFD: Cloud-native Network Function Descriptor

CSAR: Cloud Service Archive

DB: Database

ETSI: European Telecommunications Standards Institute

ISM: Information Security Management

ISP: Internet Service Provider

NWA: Network Application Servers

OCD: On-call Duty

OVA: Open Virtualization Appliance

PI: Program Increment

PM: Problem Management

QA Engineer: Quality Assurance Engineer

REM: Remote Equipment Monitoring

SAS: Statistical Analysis System

SDLC: Software Development Life Cycle

TCAS: Telia Company Application Servers

TCA: Telco Cloud Automation

TCA-CP: Telco Cloud Automation Control Plane

TKG: Tanzu Kubernetes Grid

UAT: User Acceptance Test

VAPP: Virtual Application

VCS: Version Control System

VCLOUD: VMware Cloud

VNIC: Virtual Network Interface Card

VIM: Virtual Infrastructure Manager

VMs: Virtual Machines

VNF: Virtual Network Function

VNFD: Virtual Network Function Descriptor

WOW: Ways of Working

1 INTRODUCTION

In the dynamic landscape of modern technology, the need for efficient software development and deployment processes has become paramount for enterprises striving to stay competitive. Continuous integration and continuous delivery, continuous deployment (CI/CD) pipelines play a vital role in achieving this efficiency by automating and streamlining the software delivery lifecycle. They enable organizations to rapidly deploy new features, updates, and fixes while maintaining the stability and reliability of their applications. However, as businesses continue to evolve, the demand for more agile and scalable CI/CD practices has grown.[1] This requires ongoing efforts to improve and adapt CI/CD processes to meet the evolving needs of modern businesses.

This thesis explores CI/CD pipelines within enterprise settings. It specifically examines their significance and implementation in the case study of Telia Company Application Servers (TCAS) platform within the Network Application Department at Telia Company. The purpose of the thesis is to analyze TCAS's application development and operational processes, focusing on the implementation of CI/CD pipelines. Through this analysis, identifying inefficiencies and challenges and offering practical solutions to address these challenges. The scope of the thesis encompasses various aspects, including exploring the CI/CD pipeline theory, understanding Telia's landscape, analyzing application development and operational processes, proposing solutions and implementation strategies.

The thesis begins with an introduction, outlining the research objectives, scope, and its structure. Following the introduction, Chapter 2 delves into the background of CI/CD, where fundamental concepts, tools, values, and trends are explored. Subsequently, Chapter 3 presents a comprehensive Case Study and Current Situation Analysis, offering insights into Telia Company's landscape, including its application development and operation processes. This analysis not only provides context but also highlights the specific challenges

faced by Telia, serving as a foundation for the proposed solutions. In Chapter 4, the Solution Proposal section outlines potential solutions to address the identified challenges. Each component of the solution is explained, elucidating its design, implementation strategies, and anticipated benefits. Finally, Chapter 5 concludes the thesis and discusses the implications of the research findings for organizations, limitations of the study, and future research.

2 BACKGROUND

This chapter provides an overview of Continuous Integration and Continuous Delivery/Deployment (CI/CD) pipelines, essential components in modern software development practices. The chapter reviews the relevant literature on the concepts of CI and CD, including their definitions, principles, and the value they bring to software development processes in enterprise. Additionally, it explores the tools and technologies CI/CD popular on the market and introduces their trend and growth. Through an in-depth analysis of literature and technological frameworks, this chapter aims to equip readers with a comprehensive understanding of CI/CD pipelines and the significance of implementing them within enterprise settings.

2.1 What is CI/CD

Continuous integration and continuous delivery, continuous deployment commonly shorted to CI/CD is a collection of principles and practices design which can help to optimize and accelerate the software development lifecycle.[2] These practices automate much of the manual intervention typically necessary for deploying new code into production environments. Consequently, teams can release new features and fixes more swiftly and frequently, enhancing the product's responsiveness to user demands. By implementing CI/CD, potential errors can be detected earlier in the development process, thereby minimizing downtime, and enhancing overall software quality. Additionally, CI/CD also allows for faster feedback loops with stakeholders, ensuring that the final product aligns closely with user expectations. Overall, it

serves as a fundamental practice for any team attempting for high-speed, high-quality software development. [2]

2.2 Continuous Integration (CI)

In the Continuous Integration (CI) phase in figure 1, developers commit their code to a shared repository daily. Simultaneously, the CI server on the integration build machine is actively monitoring this repository for any modifications. Once the CI server detects that new changes have been committed to the repository, the CI server retrieves the latest copy of the code from the repository and triggers a series of actions. These actions are usually defined in a configuration file or script associated with the CI server. After the completion of the build process, CI server generates feedback regarding the outcome of the build to specified project members. This feedback includes details about whether the build was successful or if there were any errors or failures encountered during the process. The CI server resumes its polling activity for changes in the version control repository.[3]

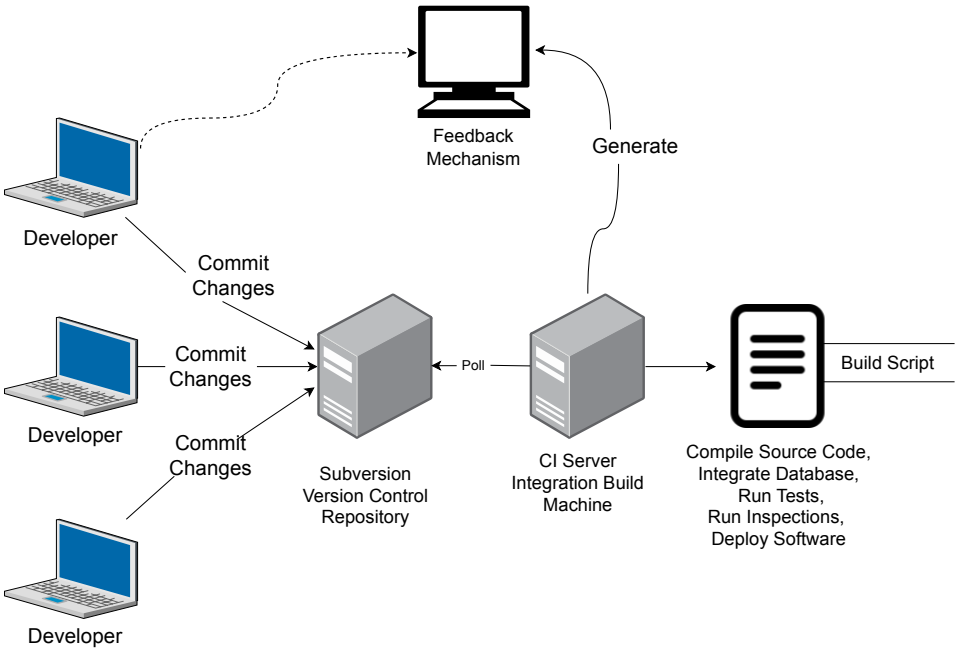

Figure 1: The component of CI systems [3]

The goal of CI is to ensure that new code added to the project avoids causing integration errors and disrupting the work of other members. With continuous integration, bugs and security issues can be identified and resolved much easier and sooner during development.[3]

By merging changes regularly and activating testing processes automated collection and validation. CI systems not only minimise the possibility of code conflict but also reduce risk, ensure the health of software, and deduction of assumptions, even when multiple developers work on the same application. Furthermore, the automation of CI results in reducing repetitive tasks, thereby decreasing the need for labour, and enabling organizations to save on costs, time, and effort.[3]

Another enormous benefit that CI systems offer a considerable advantage by reducing the waiting time between developers committing their code and receiving feedback on its quality. This eliminates the need for developers to endure long waits for assessment results, allowing them to promptly address errors and security threats as they arise.[3]

2.3 Continuous Delivery and Continuous Deployment (CD)

2.3.1 Continuous Delivery

Continuous delivery is the practice of automating the process of releasing applications and provisioning infrastructure. Once the code has been tested and built, the Continuous Delivery system controls the final stages to ensure the code is packaged with all necessary components to deploy to any environment at any time. Continuous delivery can include everything from infrastructure provisioning to application delivery to test or production environments. With Continuous Delivery, the software built can be deployed to production environments at any time. Deployments can be triggered manually or switched to Continuous Deployment where deployments are also automated.[2]

2.3.2 Continuous Deployment

Continuous deployment is the final stage of the CI/CD process, extending from continuous delivery. It can occur after the Continuous Delivery Pipeline has verified the validity and deploy ability of the changes implemented to the software. This procedure can refer to all the multiple methods that make the software available to the end user. Depending on how users are expected to install the code, deployment may mean automatically deploying in a cloud, making an update available, updating a website, or simply updating the list of available releases.[2]

However, not every new deliverable may be suitable for deployment. It might not meet the necessary quality criteria and could underperform in practical scenarios or may have security vulnerabilities that need to be addressed before deployment. In this case, manual checks, also known as User Acceptance Tests (UAT), can be incorporated. These checks would require human intervention and approval before deployment. Additionally, Continuous Deployment is not recommended for production environments. It is only suitable for lab and ref environments. This is because the production environment is the most crucial step of deployment, serving as the final stage. Declaration needs to be done manually, requiring review and approval from a competent individual or organization.[2]

2.4 CI/CD pipeline

The term “pipeline” refers to an ordered arrangement of instructions and tasks that are systematically queued for the computer processor to execute in parallel. The data processed by the previous task serves as the input for the subsequent task.

CI/CD pipeline is a series of processes designed to carry out CI/CD tasks. The pipeline enables the processing of each task in parallel rather than waiting for a task to finish and then moving onto another. [4]

2.5 CI/CD tools

CI/CD tools automate the software development lifecycle (SDLC) by streamlining the processes of continuous integration (CI) and continuous delivery/continuous deployment (CD). These tools integrate code changes from various developers, run automated tests, and deploy the software to different environments. There are many CI/CD tools available in the market such as Jenkins, GitLab CI/CD, GitHub Actions, Circle CI, Travis CI, Team City, Azure DevOps. These tools usually provide some key functionalities of CI/CD tools, including version control integration, automated builds, automated testing, deployment automation, and pipeline visualization. However, each tool still offers distinct features and cost structures, necessitating a thoughtful evaluation process. Therefore, selecting a CI/CD tool requires careful consideration of your business requirements, including specific organizational needs such as project types, resource availability, and compatibility with current tools. An ideal CI/CD tool should exhibit reliability, ease of automation, compatibility with various programming languages, and platforms [5]

2.6 Benefits of Implementing CI/CD Pipelines in Enterprises

After defining the CI/CD pipeline, the next step is to analyze its implementation advantages. CI/CD implementation in enterprises offers several benefits, including faster time-to-market, improved collaboration, enhanced quality assurance, scalability and flexibility, and risk mitigation. By automating the build, test, and deployment processes, CI/CD enables enterprises to release software updates and features more frequently, reducing time-to-market and gaining a competitive edge. Additionally, CI/CD fosters collaboration among development, operations, and quality assurance teams by providing a centralized platform for code integration, testing, and feedback, fostering transparency and accountability. Automated testing within CI/CD pipelines helps identify and address defects early in the development lifecycle, leading to higher software quality and reduced bug-fixing efforts in later stages. Moreover, CI/CD pipelines can scale seamlessly to accommodate evolving project needs, handling

increased workloads, supporting multiple environments, and integrating with diverse toolsets. Furthermore, by automating deployment processes and implementing continuous monitoring and rollback mechanisms, CI/CD pipelines minimize the risk of deployment failures and service disruptions, ensuring greater reliability and availability of applications.[6]

2.7 Trends and growth of CI/CD

CI/CD is currently undergoing significant growth and acceptance in the software development domain, with its advantages being extensively validated and resulting in widespread implementation. This transformative landscape of CI/CD is characterized by several trends. Firstly, there's the integration of predictive analytics into CI/CD pipelines, which anticipates and resolves issues beforehand, ensuring stability and quicker delivery. Additionally, enhanced automation in infrastructure management through automation tools is improving reliability and scalability in the face of increasing complexity. Moreover, there's a pervasive adoption of automated code review tools, which enhance code quality early in the development cycle, thereby balancing speed and quality. Lastly, the embracing of serverless CI/CD pipelines is simplifying infrastructure management, speeding up delivery cycles, and reducing operational overhead.[7]

3 CASE STUDY AND CURRENT SITUATION ANALYSIS

In this chapter, the case study company and context in which the thesis work was done are presented. A brief introduction, followed by ways of working as well as roles and function explanation to help viewers have a better understanding about the company landscape. Additionally, the chapter delves into the specifics of the application development process and the operational processes at Telia, offering a comprehensive understanding of its methodologies and workflows. Furthermore, a detailed analysis of the Continuous Integration/Continuous Deployment (CI/CD) pipeline within TCAS is executed, explaining the existing procedure and the obstacles encountered.

3.1 Case Study

Since 1853, Telia Company, headquartered in Sweden, has been a prominent figure in the telecommunications and Internet Service Provider (ISP) sectors. Operating across five Nordic countries - Finland, Sweden, Estonia, Latvia, and Denmark - Telia provides essential connectivity services to millions.

Within Telia the author is a part of the Network Application Development Department which designs, implements, and maintains Telia Company Application Servers (TCAS) platform and applications on top of it. This department has three Agile Teams, these teams are named Sirius, Rambo, and Titan. Each team is designated with distinct roles with Sirius focusing on the platform, while Rambo and Titan teams concentrate on various applications.

As a thesis worker, I am a part of the Rambo team. The Rambo team, especially, plays a crucial role in a spectrum of Telia's applications.

3.1.1 Agile Ways of Working

In the Network Application Department, Agile methodologies are not just a theoretical concept but are actively applied in daily operations, fostering an Agile Ways of Working (WOW) culture. This approach emphasizes collaboration, adaptability, and iterative development, aligning seamlessly with the multifaceted responsibilities of the company. The Agile methodologies are based on the Agile Manifesto foundation which has four core values supplemented by 12 principles. The Manifesto reads [8]:

“Individuals and interactions over processes and tools.

Working software over comprehensive documentation.

Customer collaboration over contract negotiation.

Responding to change over following a plan.

That is, while there is value in the items on the right, we value the items on the left more.” [8]

Each Agile methodology applies four values in different ways, but all of them rely on the philosophy to guide the software development process. The Agile WOW approach within the Rambo team is no exception [8].

Working in the team, communication, collaboration, and teamwork are very important, which aligns with the first value of the manifesto 'Individuals and interactions over processes and tools'. Regular daily stand-up meetings were scheduled throughout the week to assess the present status, review completed tasks, address any pressing matters, and outline the plans for the current and upcoming weeks. Furthermore, everyone could raise any inquiries they may have during the meeting or at any point in the chat forum.[9],[10]

Additionally, company applies many modern working software such as Jira, Confluence over comprehensive documentation.[9],[10]

Customer collaboration over contract negotiation is an important value to the team. Although the Rambo team does not engage in direct interactions with customers, they actively participate and cooperate with the customer service team to gather feedback from stakeholders throughout the development process. This structured approach allows the team to stay aligned with customer needs and requirements, rather than being strictly bound by initial contractual agreements. [9],[10]

Finally, PI Planning, known as Program Increment planning, is a crucial ritual within the Agile WOW framework, particularly in enterprises with multiple teams collaborating and interdependencies. PI planning encourages teams to embrace change and uncertainty by continuously inspecting and adapting their approach. During the planning event teams collaborate to identify dependencies, estimate work, and prioritize tasks, all while remaining flexible and responsive to shifting priorities and emerging challenges.[9],[10]

3.1.2 Roles and Functions

Additionally, the successful execution of software development projects relies heavily on the well-defined roles and functions of team members. All the teams in Telia are organized as Agile teams. The Rambo team follows the Agile methodology and operates as a virtual team. This team comprises both core team members and additional members from shared services who provide support and share their expertise. Rambo Agile Team includes three main specialty roles: The Product Owner, Scrum Master, and team members. The Product Owner holds the responsibility of defining Stories and prioritizing the Team Backlog, ensuring the execution of program priorities while maintaining conceptual and technical integrity. Meanwhile, Scrum Masters serve as servant leaders and coaches, guiding the team in following Agile processes, driving them towards PI planning targets, and facilitating sprint events. Team members are cross-functional groups defining, building, testing, and delivering increments of value within short time boxes.[11]

3.2 Application Development process in Telia

According to insights from the development team, most of the application's components is coded by Java language. They emphasized that within the telecommunications industry, particularly concerning telephony services, a unique approach to development and operation is required compared to other digital platforms such as web pages. The critical nature of telephony services demands meticulous planning and execution to ensure uninterrupted service. The development process diagram is depicted in figure 2 [12].

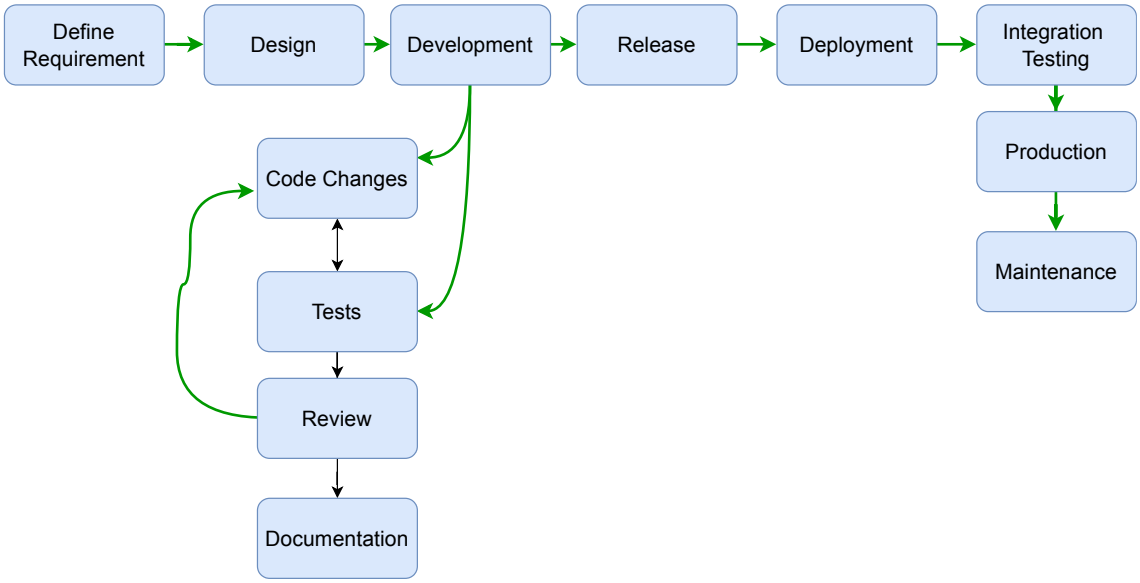

Figure 2: Telia Application Development Process [12]

As can be seen from figure 2, the development process will start with the defined requirement step which is a crucial early stage that lays the foundation for the entire project. In Telia, especially in Rambo team, the responsibility for defining requirements typically falls on some key roles such as Business Analysis, Product Owners, Product Manager, System Architects. After that, the developer team will try to understand the requirements of a feature before considering a solution.[12]

Once the solution is approved, they move to the design phase. Here, the designer analyzes how the application will be implemented, considering various components. For example, if a product has multiple services, the designer determines necessary modifications, including database and configuration changes. The level of detail varies based on product complexity and requirements. Typically led by the designer expert, developers may also participate in discussions about feasibility and estimation. The design must be finalized before the developer team starts their work.[12]

Now developers will start analysis user stories and apply the changes from the design to develop the service. The development phase includes four sub steps.

As you can see from the figure 1 that there will be code changes, tests, review, and documentation. The order of code changes and tests may vary depending on the project's needs. Typically, code changes are made first to update services, while tests are written to define desired behaviors before implementation. Various types of tests, such as unit tests or functional tests, can be used. However, functional tests are more common in telecommunication services. Functional testing evaluates software components based on specified functionalities, ensuring reliable call handling, data transmission, and network connectivity, without delving into internal structures. Review is another crucial step, where developers seek feedback from peers to ensure thoroughness and suggest improvements. Documentation is also essential, outlining the service's purpose and functionality. This can range from simple flow diagrams to detailed configurations, with updates necessary over time.[12]

If the test is approved, the reviewer marks the pull request as approved, allowing the developer to merge changes into the main branch. In Rambo team, they work directly with the main branch instead of having a separate development branch. Other teams may have development or release branches depending on workload and change tracking needs, facilitating easier management. Once changes merge into the main branch, the release step begins, tagging the service with a version number following a major.minor.patch format (e.g., 1.15.0). Patch versions increase for bug fixes, minor versions for compatible feature additions, and major versions for significant changes like database migrations. After tagging, an automated pipeline initiates, delivering the release as a zip file to a specific location in the artifact repository.[12]

Once development is approved, it's deployed in the lab. Telia operates multiple environments per country, including lab, reference, and production. Lab offers flexibility, with a message shared in the team chat once a new version is deployed. After successful lab tests, the service moves to the reference environment, mirroring production for thorough testing. Integration testing, often manual, ensures compatibility with other services, crucial as simulators may not

replicate real service behaviour accurately. This ensures testing with real data and services, including real devices.[12]

Once integration testing is done, services move to the production environment. Here, manual testing further validates system functionality, crucial to prevent live operation issues. Maintenance follows, with the developer team monitoring and addressing any production issues promptly. Detected issues prompt a return to the design phase for changes and troubleshooting.[12]

3.3 Operation process

The operational procedures in the Network Application Department combine a range of activities related to the deployment, operation, monitoring and maintaining application and system during their lifecycle. The key responsibilities and activities can be described through figure 3.

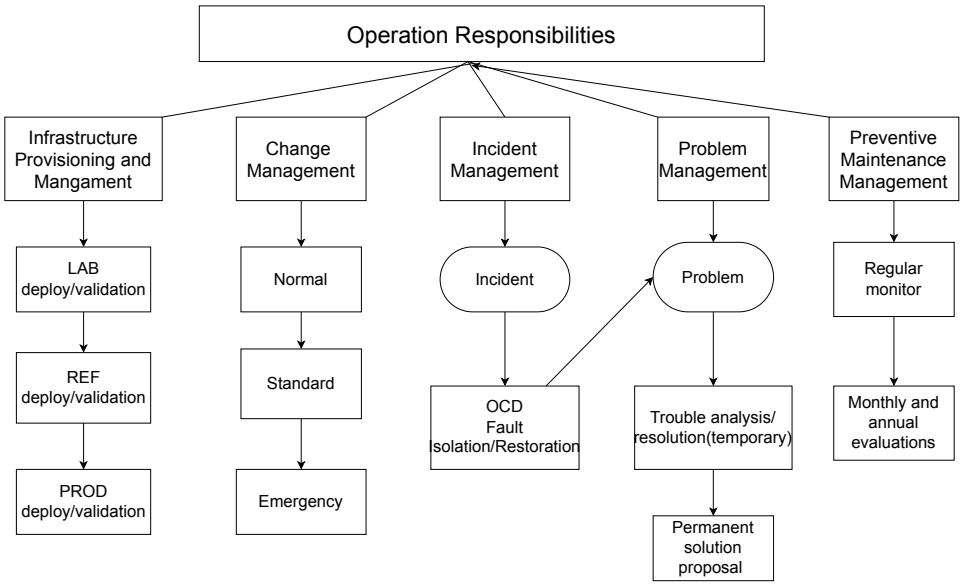

Figure 3: Operation Responsibilities in TCAS NWA

As shown in the diagram, there are five principal operation duties in NWA department include: Infrastructure provisioning and management, Change Management, Incident Management, Problem Management and Preventive Maintenance Management

Firstly, Infrastructure provisioning and management includes upgrading, patching, daily operations, and Configuration Management (CM) for all system components and the environment in a specific country. This responsibility covers both the TCAS Platform and applicable deployed Applications, whether they are common or country specific. This activity will happen in parallel with the application development process to prepare and deploy the environment for testing and operating purpose.[13]

Secondly, the primary objective of Change Management is to ensure that any modifications to IT services are effectively managed, appropriately constructed, and executed in a regulated manner that aligns with expectations of stakeholders. This process involves various procedures for realizing changes. There are different types of change requests, including Normal, Standard, and Emergency changes, each carrying different levels of risk and urgency. The Change Management process acts as a control to prevent unauthorized changes and reduce interruptions, with a focus on prioritizing critical changes for fast implementation. Overall, the focus of Change Management is to adapt to evolving business processes, minimize risks and meet stakeholders' expectations.[13]

Additionally, Incident Management is responsible for handling incidents when they occur, such as system failures or service disruptions, the operations team conducting root cause analysis, with support from the Development team if necessary. This is responsible for handling incidents during office hours. Outside of office hours, the Incident Management is handled by OCD (On-Call Duty) personnel. The responsibilities of Incident Management also include troubleshooting, fault isolation, fault resolution, and service/system recovery using temporary or intermediate solutions.[13]

Moreover, Problem Management (PM) encompasses several tasks. This includes taking ownership of PM tickets associated with Information Security Management (ISM). Additionally, it involves assigning these tickets to the appropriate team within the NWA department. Furthermore, it entails coordinating investigations to identify the root cause of the problem. In cases where the problem is complex and requires investigation across different domains, ISM may establish a specialized task force team. The Problem Management process can involve the support of the Development team if needed. Furthermore, fault replication in a test environment is conducted, often with assistance from the vendor if necessary. Problem Resolution and a proposal for permanent resolution are also part of Problem Management. Additionally, a comprehensive strategy is formulated to decrease the occurrence of the error until a long-term solution is implemented.[13]

Lastly, Preventive Maintenance Management involves the regular monitoring of system and service performance daily. Additionally, Information Security Management conducts monthly and annual evaluations on trends of system and service performance.[13]

3.4 Detail Analysis of CI/CD pipeline implementations in TCAS

In this section, the CI/CD pipeline within TCAS NWA is explored, investigating its integration and workflow in both application development and operational processes. By examining this aspect, valuable insights are obtained regarding the company's CI/CD pipeline landscape, the challenges it faces, and potential strategies for enhancement.

In TCAS NWA, the CI/CD pipeline is presently exclusively integrated into the application development process. The expansion into the Operations phase, particularly for infrastructure provisioning and operational monitoring of

applications, is currently being developed. The diagram provided in figure 3 highlights the CI/CD pipeline utilized in the application development procedure.

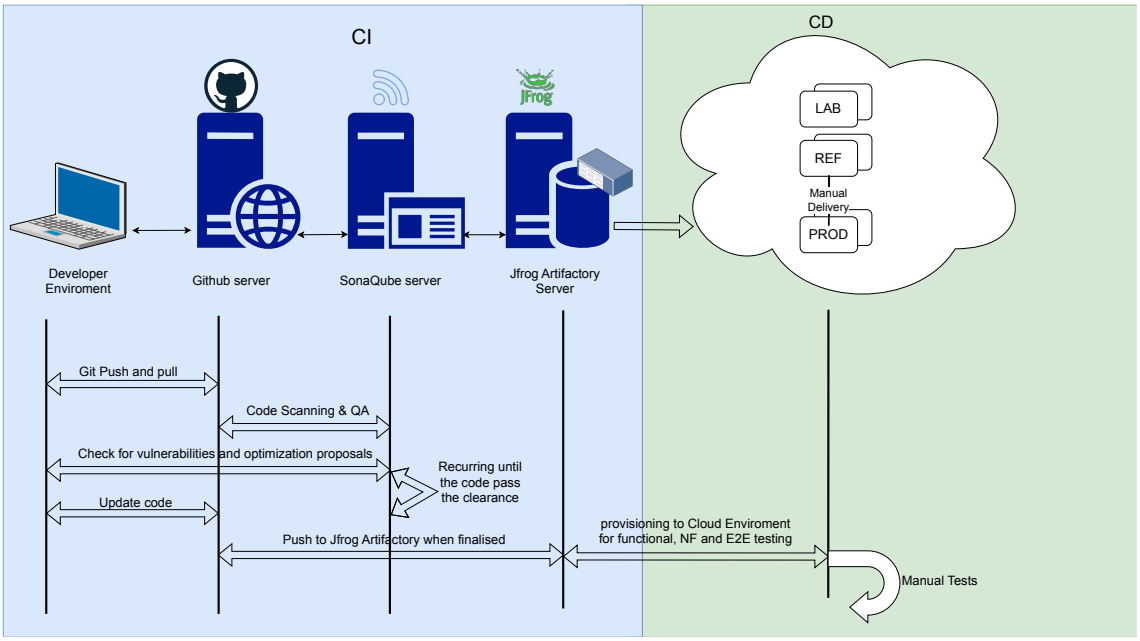

Figure 4: CI/CD in TCAS NWA (Rambo) [13]

As illustrated in Figure 4, the CI workflow begins with developers writing and committing code changes to GitHub. This step highlights the initial phase where developers make modifications to the codebase locally, ensuring that changes are saved along with descriptive messages.

Following the code commits, GitHub Actions comes into play, executing, deploying, and testing the code automatically. This phase emphasizes the automation aspect of the CI/CD process, showcasing how predefined actions are triggered upon code changes, streamlining the development pipeline. GitHub action will integrate with SonarQube to review code automatically. SonarQube is an automation review tool which helps to analyse source code either during the build process or when committing changes. Its primary functions include ensuring code quality, addressing code smells, enhancing maintainability and readability, and detecting security vulnerabilities.

Additionally, it also measures and manages technical debt to ensure long-term project sustainability.

The subsequent step involves the analysis of test results. If the code passes all tests, it proceeds to the next phase seamlessly. However, if any tests fail, it indicates potential issues with the code, necessitating a return to the developer for fixes. Upon identifying and addressing the issues, the developer updates the code locally and re-commits the changes to GitHub, initiating another cycle of GitHub Actions execution, deployment, and testing. This iterative process highlights the iterative nature of CI, emphasizing continuous improvement and refinement of the codebase.

If the code passes all tests successfully, it is published to JFrog Artifactory, ensuring that the stable version of the code is securely stored and ready for deployment to production or other environments. This last step underscores the importance of artifact management in ensuring the reliability and accessibility of the codebase.

Once the code has been tested and built, the Continuous Delivery system controls the final stages to ensure the code is packaged with all necessary components to deploy to any environment at any time. Continuous delivery can include everything from infrastructure provisioning to application delivery to lab, ref, or production environments for Functional, Non-Functional, and End-to-End testing.

3.4.1 Challenges analysis

The Continuous Delivery/Deployment in TCAS NWA is a work in progress. While automation tools have been integrated into various aspects of infrastructure provisioning, significant manual effort persists across multiple stages. Additionally, the application deployment in company does not require several important manual tests before it can be deployed in production since for

the characteristics of telephony application. The current infrastructure provisioning process can illustrate through figure 5.

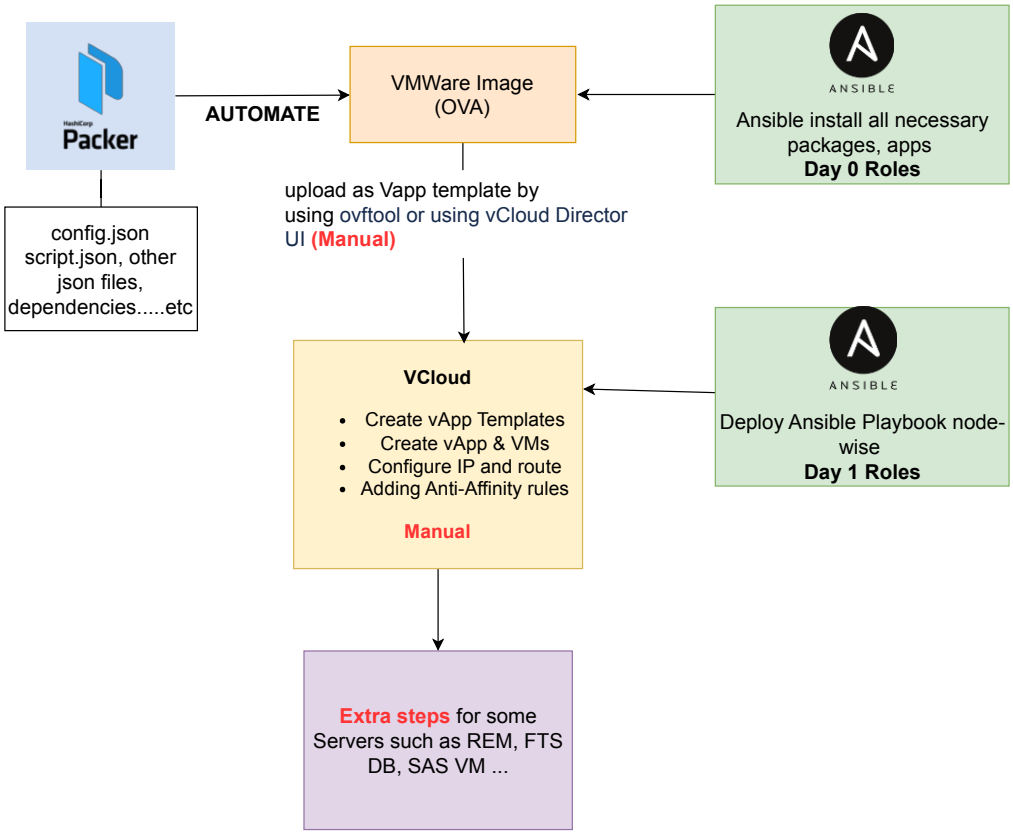

Figure 5: Current Infrastructure provisioning process in TCAS NWA

The workflow described in figure 5 comprises five main steps. Firstly, Packer is utilized to automate the creation of machine images across multiple platforms. Through a JSON configuration file, scripts, and dependencies, Packer generates an OVA (Open Virtualization Appliance) image, serving as a template for virtual machines (VMs). [13]

Subsequently, Ansible is employed to provision the OVA image. Utilizing playbooks, which are scripts defining tasks to execute on remote machines, Ansible installs necessary packages and applications onto the OVA image, typically involving roles tagged as "Day 0." Following this, the templates are manually uploaded to vCloud as vApp templates, facilitated either through the

vCloud Director UI. vCloud is a cloud management platform that facilitates the deployment and management of virtualized resources. Within vCloud, various manual steps configure the VMs and vApps, including adding networks/vSwitches, configuring CPU resources, and setting up network interfaces. [13]

Once deployed, Ansible is again used to customize the VMs further, deploying playbooks with roles tagged as "Day 1" to install additional packages tailored to each node's specific needs. Additional configuration steps may be required for specific server types such as REM, databases (DB), and SAS VMs, involving further setup specific to each server type's requirements.[13]

After analyzing TCAS's infrastructure provisioning process, it's evident that all servers in TCAS are deployed by using virtual machines. Additionally, tasks like uploading templates to vCloud, configuring VMs and vApps, adding networks/vSwitches, and setting up virtual network interface cards (vNICs) and CPU resources still heavily rely on manual intervention. Moreover, the process involves additional steps like configuring hostnames, network interfaces, and default routes. This leads to inefficiencies and potential errors and time-consuming in infrastructure deployment. This can result in delays in onboarding applications, leading to a slower time to market, a loss of competitive advantage, and customer dissatisfaction. To address these inefficiencies and expedite infrastructure deployment, the following chapter explores my solution proposal.

4 SOLUTION PROPOSAL AND EVALUATION

This chapter presents a comprehensive solution designed to address the challenges and opportunities inherent in TCAS. Throughout the chapter, a detailed exposition of each component of the solution is provided, elucidating its design, implementation strategies, and anticipated benefits. Additionally, insights into the evaluation process employed to assess the effectiveness and feasibility of the proposed solution are offered. By the chapter's conclusion,

readers will have gained a comprehensive understanding of the solution proposal and its potential to drive transformative change within TCAS.

4.1 Solution Proposal

After evaluating all the TCAS servers, I propose a solution that involves classifying the servers into two distinct categories: stateful servers and stateless servers. The stateful servers will be implemented as Virtual Network Functions (VNF), while the stateless servers will be deployed as Cloud-native Network Functions (CNF). Furthermore, Telco Cloud Automation serves as the management and orchestration toolset, streamlining application and service deployment and operation within the Telco Cloud environment. This strategic approach aims to minimizing manual intervention, enhance operational efficiency and accelerate the time deployment of servers within the TCAS landscape.

4.1.1 Understand Stateful and Stateless servers.

A stateful server is the server that maintains information about the state of each client session. This means it keeps track of past interactions and remembers the context of each session. Stateful servers store session data on the server side and associate it with a particular client session. These servers typically require less overhead in terms of data transmission because they only need to send updates when the state changes. Examples include Database servers, Application servers, Management servers. Database servers store state data about clients' previous queries or transactions, while Application servers may retain information about the current state of the application a client is using. Management servers might also maintain state about devices and system resources in the network infrastructure. [14]

On the other hand, a stateless server treats each request from a client as an independent transaction that does not rely on past interactions. Stateless servers do not store client session data on the server side. Instead, they rely on

the client to include all necessary information with each request. These servers are often more scalable and easier to manage because they do not need to maintain session state. Examples include RESTful APIs, HAproxy serves, Provisioning gateways. RESTful APIs typically handle requests and responses without retaining information about previous interactions. HAproxy serves as a load balancer that forwards requests without tracking session data, and Provisioning gateways handle requests for network configuration or resource allocation without storing client session information. [14]

4.1.2 Understand VNF and CNF

VNF stands for Virtual Network Function. It refers to network services or functions that are decoupled from the underlying hardware and implemented in software that runs on virtualized infrastructure. These virtualized functions are designed to replace traditional, dedicated hardware appliances, such as routers, firewalls, load balancers, and more.[9] VNFs are implemented as virtual machines (VMs) and are the current technology that company applies to provision infrastructure. The deployment process presented in section 3.4.2.

Cloud-native network functions (CNFs) represent a specialized form of virtualized network functions designed to operate within containers. Unlike traditional virtual machines (VMs), containers offer a lightweight solution that enables users to package software, including applications, functions, or microservices, along with all necessary files for execution. One significant advantage of containers over VMs is their efficiency in resource utilization, as they do not require a guest operating system or hypervisor. This efficiency allows to package software without concerns about the Operation System. This efficiency translates to faster start-up times and more efficient use of infrastructure resources, ultimately improving scalability and flexibility in network function deployment.[15],[16]

4.1.3 Telco Cloud Automation

VMware Telco Cloud Automation (TCA) is a tool already approved in Telia, a unified orchestrator that automates the management and coordination of resources within the Telco Cloud environment. What makes TCA unique is its ability to work with both virtual machine (VM) and container-based infrastructures.[17]

TCA not only automates the onboarding process of applications and resources but also orchestrates them seamlessly, from core data centers to edge locations, as well as from private to public clouds. This creates a diverse and flexible service-delivery foundation for telecommunications service providers.[17]

In essence, TCA is a comprehensive management and orchestration toolset that optimizes the deployment and operation of applications and services within the Telco Cloud environment. [17]

4.2 Solution Design

The integration of TCA to the Network Function Virtualization Infrastructure (NFVI) and Virtual Infrastructure Managers (VIMs) for managing VNF and CNF is displayed in the figure 6.

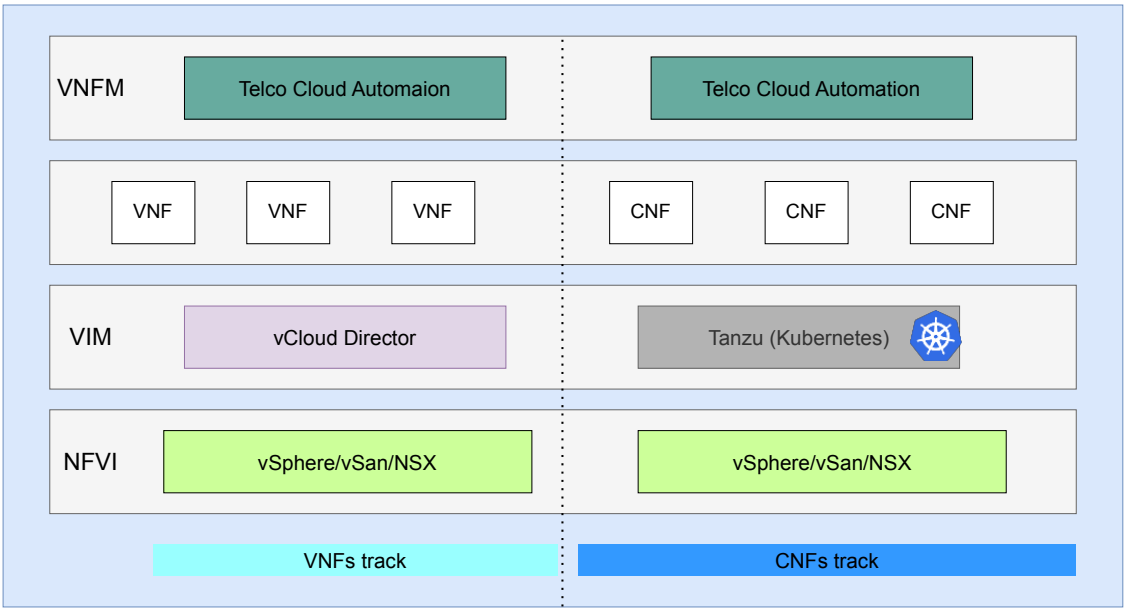

Figure 6: Solution design diagram

As illustrated in the figure 6, it showcases two main tracks: Virtual Network Functions (VNF) and Containerized Network Functions (CNF). Both tracks share a common foundation. The layer at the bottom is the Network Function Virtualization Infrastructure (NFVI), which provides the essential resources to run both VNFs and CNFs. It comprises three core components:

- vSphere: A core component that manages the lifecycle of virtual machines (VMs). Within the VNF track, vSphere serves as the foundation for deploying VNFs as VMs. Additionally, vSphere provides the necessary infrastructure to support and manage Container Network Functions (CNFs) alongside VMs. [18]

- vSAN: This software-defined storage solution pools the local storage resources of the physical servers hosting the VMs. This pooled storage provides a shared and scalable platform for both VNFs and CNFs.[18]

- NSX: This network virtualization platform virtualizes network services such as switching, routing, and security. NSX is crucial for enabling

communication between VNFs and CNFs deployed on the Telco Cloud infrastructure.[18]

The next layer is Virtual Infrastructure Managers (VIMs). This layer manages the NFVI resources. It includes:

- vCloud Director: A VIM that offers a self-service portal for provisioning, managing, and monitoring VMs deployed as VNFs. vCloud Director simplifies administration and empowers users to manage their VNF resources.[18]

- Tanzu CaaS (Containerized as a Service): While not always used, Tanzu CaaS is a potential VIM. Tanzu offers a collection of tools like Tanzu Kubernetes Grid (TKG) that helps to set up and manage Kubernetes clusters across different environments. Kubernetes is an open-source system designed to automate the deployment, scaling, and management of containers. Since CaaS relies on Kubernetes for container orchestration, Tanzu provides the foundation for running CaaS solutions. [19]

Finally, the layer sits on top of the entire architecture is Telco Cloud Automation (TCA). This is responsible for automating the deployment and management of both VNFs and CNFs. TCA-Control Plane (TCA-CP) acts as the virtual infrastructure backbone, connecting Telco edge, aggregation, and core sites. TCA-CP enables workload placement across clouds and supports various Virtual Infrastructure Manager (VIM) types, including VMware vCenter Server, VMware vCloud Director, VMware Integrated OpenStack, and Kubernetes. Telco Cloud Automation interfaces with TCA-CP to interact with VIMs to leverage NFVI resources to streamline network function provisioning and lifecycle management. The VIMs are cloud platforms such as vCloud Director, vSphere, Kubernetes cluster, or VMware Integrated OpenStack.[17]

4.3 VNF onboarding with TCA implementation strategy

To gain a deeper understanding of how TCA serves as a comprehensive deployment platform for automating VNF deployment, it is essential to explore the potential of automating VNF deployment with TCA. Figure 7 illustrates the automation of VNF deployment with Telco Cloud Automation.

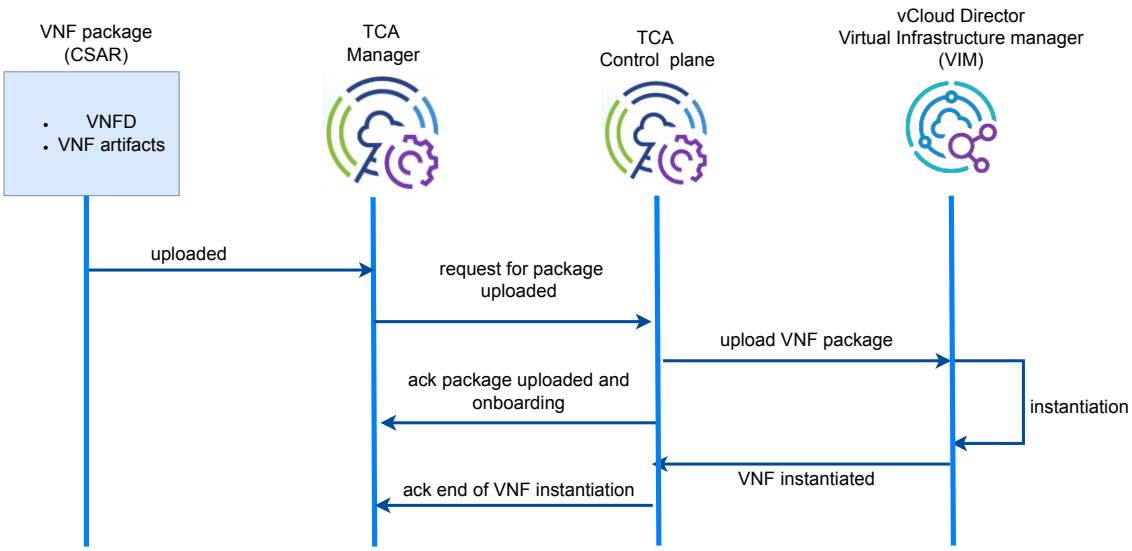

Figure 7: VNF onboarding and deploying with TCA.

As can be seen from the diagram, the VNF onboarding process begins with preparing the VNF package known as CSAR (Cloud Service Archive) defined by ETSI and following YAML specifications. This package encapsulates all the necessary information for deploying and managing the lifecycle of a VNF. A CSAR typically includes two key components.

- Virtual Network Function Descriptor (VNFD): The foundation for automated network configuration lies in VNFDs. These standardized descriptors detail the characteristics of a VNF, including its network connectivity requirements. VNFDs specify internal connection points within the VNF and may also reference external network connections.[20]

- VNF Artifacts: These artifacts encompass the actual software components and dependencies required for the VNF to function.

Upon reviewing the figure 6, it becomes clear that once the VNF package is uploaded to the TCA manager, a request is made for it to be transferred to the TCA control plane. Subsequently, the TCA control plane proceeds to transfer the VNF package to the VIM, specifically to the vCloud Director in this scenario. After a successful onboarding, the vCloud Director directs the system to instantiate the VNF, involving deployment and resource provisioning. Ultimately, upon instantiation of the VNF, the TCA control plane notifies the TCA manager of completion, marking the end of the process.

4.3.1 Advantages when onboarding VNF with TCA

The process of VNF onboarding and deploying with TCA brings numerous advantages which illustrate through the figure 8 below.

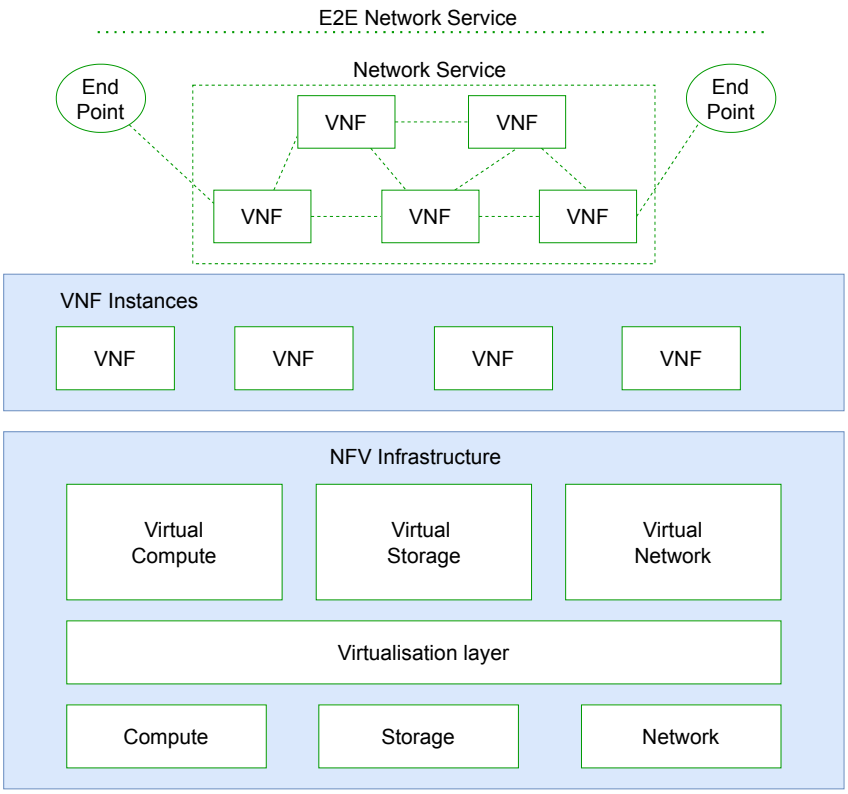

Figure 8: End to End network service with VNFs [21]

TCA revolutionizes VNF deployment, as can be seen from figure 7, it streamlines infrastructure provisioning through end-to-end automation, ensuring

standardization and consistency across the network. Additionally, during VNF instantiation, TCA leverages the information within the VNFD to automatically create virtual networks for internal communication between the VNF's Virtual Network Units (VNUs) such as firewalling, routing, load balancing, etc. This eliminates the need for manual configuration, reducing errors and streamlining the deployment process. Furthermore, TCA automates network creation between multiple VNFs, as it manages the creation and configuration of all VNF network connections within a single platform. This simplifies the overall management of network services with interconnected VNFs, providing a centralized view and control point.

4.4 CNF onboarding with TCA implementation strategy

This strategy is based on [22], with a container image creation CI/CD pipeline that I designed.

Prior to creating a Cloud-native Network Function (CNF), it is essential to have both a Tanzu Kubernetes Grid (TKG) management cluster and a Tanzu Kubernetes Grid (TKG) workload cluster established within VMware Telco Cloud Automation. The workload cluster serves as the designated environment for deploying the CNFs. Creating a workload cluster involves setting control nodes and worker nodes. [22]

Setting control nodes involves configuring and deploying the essential components responsible for managing the state of a Kubernetes cluster. These control nodes in figure 9, comprising elements such as the scheduler, controller-manager, and kube-API, play a critical role in orchestrating and coordinating workload deployment and management across the cluster. In the context of TCA, the TKG control nodes are specifically designated for decision-making tasks and are not utilized for running actual workloads; instead, worker nodes handle workload execution. Additionally, control nodes typically only have access to the management network, ensuring secure communication and management of the cluster infrastructure.[22]

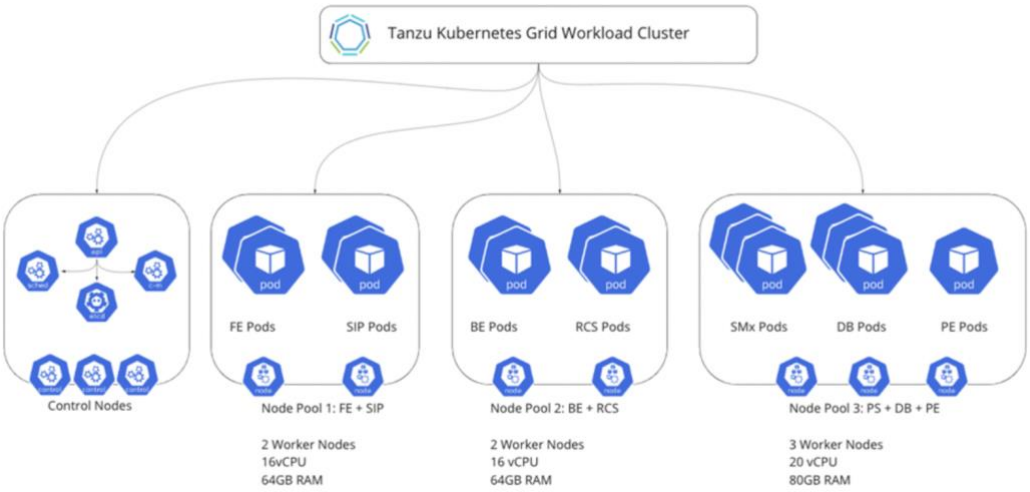

Figure 9: Tanzu Kubernetes Grid Workload Cluster example [16]

On the other hand, worker nodes act as the workhorses of the cluster. This is where containerized applications and pods are scheduled and executed. The figure 9 showcases a workload cluster designed with multiple worker nodes grouped into node pools. Each node pool offers a specific configuration tailored to the needs of the pods it accommodates.[22]

Cloud-native Network function onboarding incorporates multiple elements. Not all these elements are bound to TCA [22].

- CSAR (Cloud Service Archive) holds the CNF descriptor (CNFD) and other components specific to the application and is uploaded to TCA.[22]

- Helm chart is a standardized way of packaging Kubernetes manifests together in a structured format. These are uploaded to Harbor.[22]

- Values.yaml is used during instantiation and overrides the default configuration from the HELM chart. This allows a common HELM configuration to be prepacked, but deployment specifics are supplied during deployment through this file.[22]

- Container Images are docker- compliant container images that are tagged and uploaded to a Harbor deployment.[22]

To create a container image, I proposed a CI/CD pipeline with Tanzu built services is proposed in the figure 10 below.

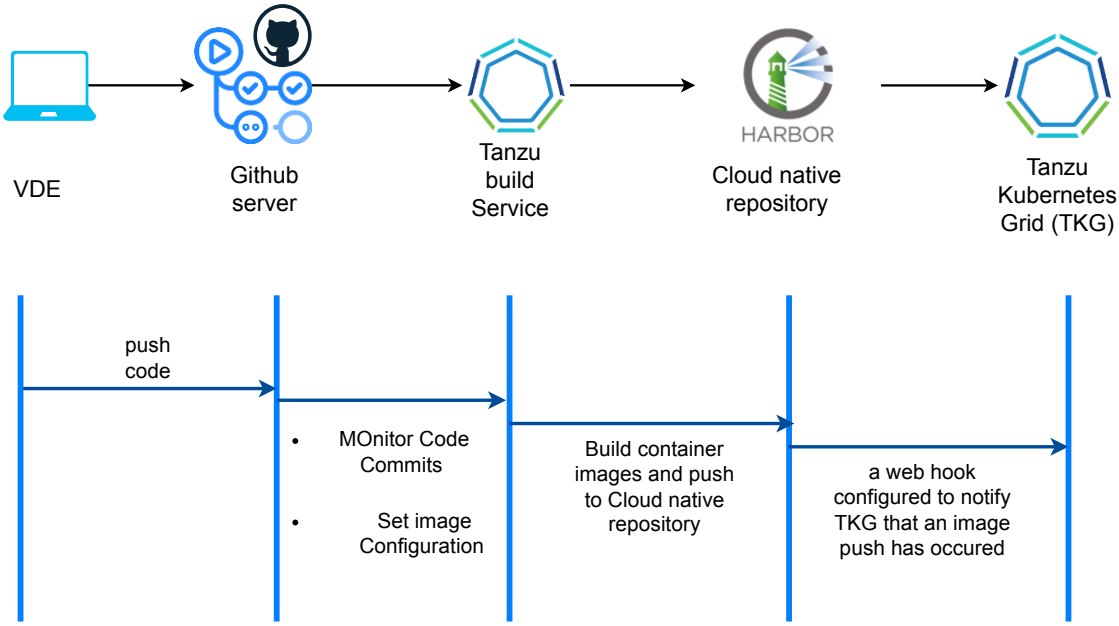

Figure 10: Tanzu Build Service CI/CD pipeline for Image Creation

As can be seen from Figure 9, the developer commits code to the GitHub server. Once this merge is performed, GitHub Action detects the new code and instructs the Tanzu Build Service to update the configuration of a specific image. Subsequently, the image is built through Tanzu Build Service and then pushed to the designated container registry, Harbor. Harbor is a VMware product, serves as a secure and scalable container registry solution designed for enterprise environments. It provides features such as image vulnerability scanning, access control, and replication. After the image is built through Tanzu Build Service and then pushed to Harbor and a webhook in Harbor notifies Tanzu Kubernetes Grid of the image push.

Figure 11 provides a visual representation of the automated onboarding process for Cloud-Native Network Functions (CNFs) using VMware Telco Cloud Automation.

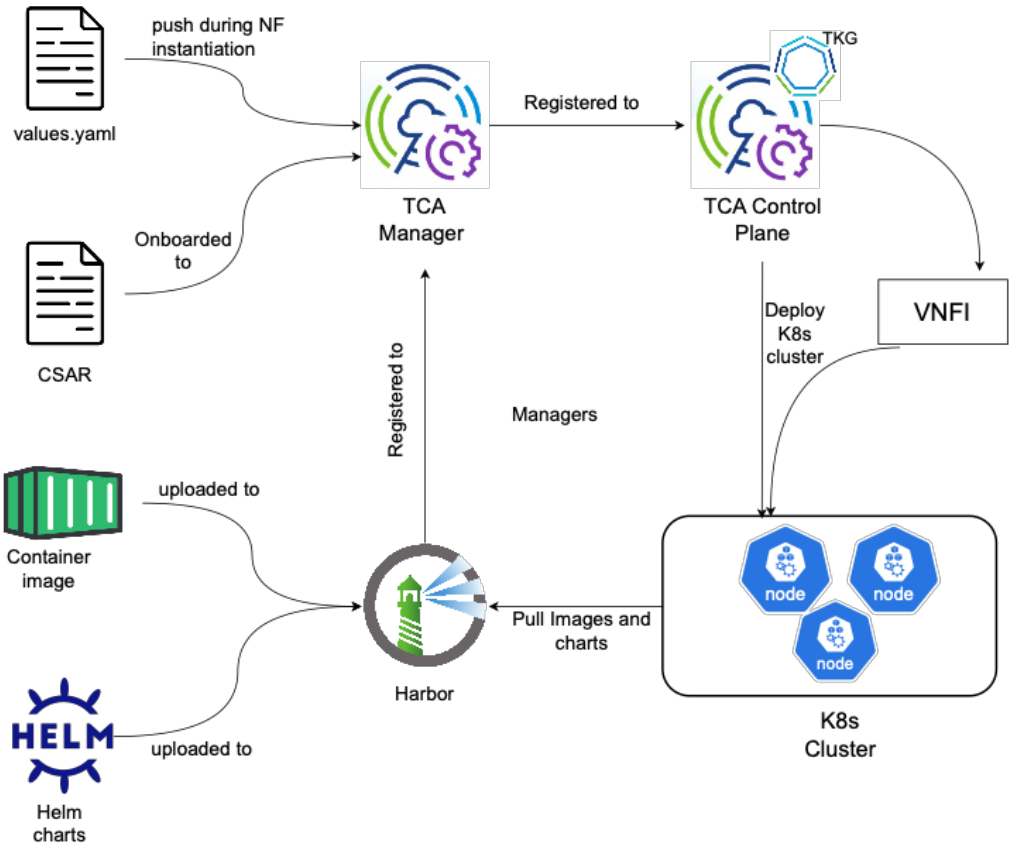

Figure 11: Automated CNF Onboarding with VMware Telco Cloud Automation [21]

As illustrated in the diagram, when performing CNF instantiation, the TCA manager acquires the values.yaml file and CSAR package, which contain essential configuration details for deployment.

To deploy Kubernetes clusters, the TCA manager establishes a connection with the central TCA control plane in this context is Tanzu Kubernetes Grid (TKG). TKG manages and communicates with VNFI to deploy a K8s cluster.

Both the Helm chart and the container images undergo upload to the container image registry, Harbor. These container images are built through Tanzu Build Service, as illustrated in figure 9. This centralized registry serves as a repository for these artifacts. Upon the push of the Helm charts and container images to

Harbor, a notification is relayed to the TCA manager. This communication ensures that the TCA manager is aware of the availability of these artifacts for deployment.

Finally, the deployed K8s cluster will pull Helm charts and container images uploaded to automatically manage and deploy nodes.

4.4.1 Advantages when deploying CNF with TCA

Additionally, deploying stateless servers with Cloud-native Network Functions (CNFs) provides several advantages over traditional Virtual Machine (VM) deployments. CNFs leverage lightweight containers, resulting in faster deployment and scaling processes due to reduced resource consumption. Containers share the host operating system, optimizing resource allocation and leading to significant time savings. They also ensure a consistent environment across different deployment stages, facilitating seamless migration and simplifying management processes. [15]

In addition, TCA integrates with VMware Tanzu to enhance automation capabilities, streamlining container orchestration tasks. Given that Container-as-a-Service (CaaS) heavily relies on Kubernetes for container orchestration, Tanzu serves as the foundational platform for running containers or any Kubernetes. By leveraging Tanzu's capabilities, operators gain access to advanced features such as automated scaling, rolling updates, and service discovery. It empowers efficiently management CNFs and enhances the resilience and performance of their network infrastructure.[19],[23]

5 CONCLUSION

5.1 Summary of findings

In this thesis, I delved into researching the CI/CD pipeline, covering definitions, principles, tools, their values, and future trends. Furthermore, I also investigated

the application development and operational processes within TCAS, proposing solutions to bolster efficiency and agility in the operational procedure. Throughout the research, several key findings have emerged.

Firstly, an in-depth exploration of the CI/CD pipeline provided invaluable insights into its processes, tools, trends, and growth. This exploration helped confirm my understanding and laid the foundation for designing a robust CI/CD solution proposal.

Secondly, acquiring a profound understanding of Telia's landscape, working methods, including the roles and functions of team members, provided numerous benefits in the thesis working process, from data collection to understanding the company’s current situation and obstacles.

Furthermore, the analysis of TCAS's current application development and operational processes uncovered significant opportunities for improvement. Existing infrastructure provisioning models were found to lack the requisite flexibility and agility, leading to delays, inefficiencies, and challenges in aligning with the development process. To address these challenges, the research proposes a solution that involves categorizing servers within TCAS into two main types: stateful servers for deployment as Virtual Network Functions (VNFs) and stateless servers for deployment as Cloud Native Functions (CNFs). This classification aims to minimize manual intervention, enhance operational efficiency, and expedite server deployment within the TCAS infrastructure. The solution comprises component explanation, solution design, and implementation strategies. Furthermore, in the CNF deployment implementation strategy, the study introduces a CI/CD pipeline for creating container images for CNFs.

5.2 Limitations of the study

While this research has provided insights and practical recommendations, it is essential to acknowledge its limitations. The study was conducted within a

limited timeframe, which may have constrained the depth and breadth of the analysis. Certain aspects, such as the long-term impacts or scalability of the proposed solutions, may require longer-term observation and evaluation to fully understand their effectiveness. The study may not have captured the perspectives of all relevant stakeholders within Telia, including end-users, middle management, or external partners. Incorporating diverse viewpoints could have provided a more comprehensive understanding of the challenges and opportunities related to CI/CD implementation.

Moreover, the research mostly focused on the technical parts of the CI/CD pipeline and did not investigate extensively into security and change management. Additionally, with technology constantly updating, it is important to regularly review and revise the plan and suggestions. This includes the ongoing development of containerization technologies, Telco Cloud Automation, and CI/CD practices necessitates ongoing research and adaptation to ensure the relevance and effectiveness of the proposed recommendations. Therefore, the practical implementation of the findings to real-world scenarios may require further exploration and validation in the ever-evolving landscape of telecommunications and software development.

5.3 Future work and research

While the research findings are promising, there are several areas requiring further investigation. Firstly, concerning the operational processes within TCAS, additional inquiry is warranted to address the limitations identified in this study. Research efforts could delve deeper into refining the proposed solution, focusing on scalability, resilience, and adaptability to evolving network demands.

Additionally, it is necessary to evaluate the impact of suggested proposals on project success indicators. These indicators may include cost savings, technological compatibility, and time-saving benefits. This evaluation could be explored through quantitative analysis.

Furthermore, exploring strategies for the practical implementation of solutions in real-world scenarios involves continuing to investigate the operationalization of proposed tools and technologies. It also entails improving the CI/CD pipeline with CNF. Additionally, it requires considering factors such as organizational readiness, resource availability, and potential deployment challenges.

REFERENCES

1 Continuous Integration vs. Continuous Delivery vs. Continuous Deployment, 2nd Edition. O'Reilly Media. Available at: https://learning.oreilly.com/library/view/continuous-integration-vs/9781492088943/ch01.html#idm45972890245720. Published 2017. Accessed March 5th,2024.

2 Team RH. What is Ci/CD? Red Hat - We make open source technologies for the enterprise. https://www.redhat.com/en/topics/devops/what-is-ci-cd. Published 2022. Accessed March 3rd, 2024.

3 Continuous Integration: Improving Software Quality and Reducing Risk. O'Reilly Media. Available at: https://learning.oreilly.com/library/view/continuous-integration-improving/9780321336385/ch02.html#ch02lev1sec1. Published 2007 Accessed March 5th,2024.

4 What does pipeline mean? Techopedia. Available at: https://www.techopedia.com/definition/5312/pipeline. Published 2017. Accessed March 28th ,2024.

5 How to Choose a CI/CD Tool. JetBrains TeamCity Blog. Available at: [https://blog.jetbrains.com/teamcity/2023/08/how-to-choose-cicd-tool/]. Accessed April 30th, 2024.

6 Benefits of CI/CD. JetBrains TeamCity. Available at: [https://www.jetbrains.com/teamcity/ci-cd-guide/benefits-of-ci-cd/]. Accessed April 30th, 2024.

7 CI/CD Trends and Predictions for 2024. TMCNet. Available at: [https://it.tmcnet.com/topics/it/articles/2023/12/05/457907-cicd-trends-predictions-2024.htm]. Accessed April 30th, 2024.

8 21 Lean-Agile Mindset. Scaled Agile Framework. Available at: [https://v5.scaledagileframework.com/lean-agile-mindset/]. Accessed April 20th, 2024.

9 Team Performance in Software Development: Research Results versus Agile Principles. IEEE Xplore. Available at: [https://ieeexplore-ieee-org.ezproxy.metropolia.fi/document/7498535]. Accessed April 20th, 2024.

10 Telia Company. 2024. Internal document.

11 Telia Company. 2024. Internal document.

12 Interview with Telia Development Team. Telia Company. 2024. Internal document

13 Telia Company. 2023. Internal document.

14 Stateful vs. stateless applications. Red Hat. Available at: [https://www.redhat.com/en/topics/cloud-native-apps/stateful-vs-stateless]. Accessed April 20th, 2024.

15 What is Cloud Native Network Functions (CNF)? ZenArmor. Available at: [https://www.zenarmor.com/docs/network-basics/what-is-cloud-native-network-functions-cnf#:~:text=CNFs. Accessed 20th, 2024.

16 VNF and CNF: What's the difference? Red Hat. Available at: [https://www.redhat.com/en/topics/cloud-native-apps/vnf-and-cnf-whats-the-difference]. Accessed April 20th, 2024.

17 VMware Telco Cloud Platform 5G Edition Reference Architecture Guide. VMware Docs. Available at: [https://docs.vmware.com/en/VMware-Telco-Cloud-Platform/2.7/telco-cloud-platform-5G-edition-reference-architecture-guide-27/GUID-C19566B3-F42D-4351-BA55-DE70D55FB0DD.html#:~:text=Telco Accessed April 30th, 2024.

18 VMware Telco Cloud Infrastructure Datasheet. VMware. Available at: [https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/vmw-telco-cloud-infrastructure-datasheet.pdf]. Accessed April 6th, 2024.

19 VMware Tanzu. Tanzu by VMware. Available at: [https://tanzu.vmware.com/tanzu]. Accessed April 30th, 2024.

20 Understanding Virtual Network Function Descriptors. Cisco. Available at: [https://www.cisco.com/c/en/us/td/docs/net_mgmt/elastic_services_controller/4-4/user/guide/Cisco-Elastic-Services-Controller-User-Guide-4-4/understanding_virtual_network_function_descriptors.pdf]. Accessed April 30th, 2024.

21 NSR White Paper: Satellite Telecom Network Integration v4.1. NSR. Available at: [https://www.nsr.com/wp-content/uploads/2022/03/NSR-White-Paper-Satellite-Telecom-Network-Integration-v4.1.-Lower-File-Size.pdf]. Accessed April 30th, 2024.

22 Amin E. Kubernetes Cluster Creation and CNF Onboarding. LinkedIn. Available at: [https://www.linkedin.com/pulse/kubernetes-cluster-creation-cnf-onboarding-eslam-amin-ezurf/]. Accessed April 10th, 2024.

23 VMware Telco Cloud Automation. VMware Telco Cloud. Available at: [https://telco.vmware.com/products/telco-cloud-automation.html]. Accessed April 5th, 2024.