



THESIS

WEB-BASED DEVELOPMENT ENVIRONMENT FOR TACTICOS APPLICATIONS
IMPLEMENTED ON A CLOUD-NATIVE PLATFORM.

Year	2022/2023		
Educational institution	Saxion University of Applied Sciences		
Module	Graduation HBO-ICT		
Involved teacher	Herman Voortman	Involved students	Max de Haas
Document version	1.0	Number of pages	98
Realization date	20-01-2023	First realization	19-09-2022

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

VERSION MANAGEMENT

Version	Changes	Date
0.1	Added 1. Introduction and 2. Techniques and Methods	19-09-2022
0.2	Set up problem analysis	10-10-2022
0.3	Set up Needs analysis	19-10-2022
0.4	Set up Context analysis	28-10-2022
0.5	Set up Literature study	01-11-2022
0.6	Set up Design/testing	29-11-2022
0.7	Adding all the appendixes	23-12-2022
0.8	Performing quality checks/changing small things	27-12-2022
0.9	Finished concept version	07-01-2023
1.0	Change feedback Saxion	16-01-2023

Table 1 Version management

Name	Role	E-mail
Herman Voortman	Supervisor/teacher from Saxion	h.w.voortman@saxion.nl
Willy Boenink	Supervisor/Software engineer from Thales Nederland B.V.	willy.boenink@nl.thalesgroup.com
Max de Haas	Graduation student	448590@student.saxion.nl max.dehaas@nl.thalesgroup.com

Table 2 Contact information supervisors and graduate

MANAGEMENT SUMMARY

Many companies have recently started implementing cloud-native applications, and Thales has done the same. These applications are currently being used in certain parts of the company. With the development environment missing. The goal of the assignment is to do research on how the development environment can be placed on a Cloud-native platform. The development environment is divided into 4 parts for assignment purposes, these are the development part, the version control system, the build/test system and an artifactory.

Research has been done help design a cloud-native development environment and ensure that all cloud-native applications used communicate with each other and provide security, manageability, observability and scalability. The research method used during the assignment is “design research”, the design research method is based on scientifically proven research and results in design(s) and an advisory. The advisory and design are based on the requirements, the design principles and conditions.

The main question the research and where the focus is on is:

“How could a web-based development environment for TACTICOS applications be implemented on a cloud-native platform?”

Besides the main questions there are multiple sub-questions, these sub-questions will result into the answers for the main question. The research is split up into multiple parts: a problem analysis, needs analysis and a context analysis. These parts deliver a view of the problems, gather the stakeholders and note their needs and define the conditions of for the design.

The following conclusion has been concluded and summarized in these results.

A Cloud-native infrastructure can help a company save on its resources, using less resources than a physical application. The scalability makes it easier to deploy more instances of an application without configuring them. This can be done with Kubernetes as a platform and then the choice can be made on what development solution can be implemented. The tests have been done with GitLab and the current application set build on a Kubernetes platform instead of the virtual machine it runs right now.

The outcome of the tests was that a web-based development can be deployed on a Cloud-native platform without any problems.

With these test results and the capabilities of Cloud-native, the following advice was determined: Thales should make the step to a Cloud-native development environment as it can only help them better maintain their system, more flexibility, more scalability and save on resources. With

1 TABLE OF CONTENTS

Abbreviations and Glossary.....	5
Preface	6
1 Introduction	7
2 Techniques and methodolgies	9
2.1 Design Research.....	9
2.2 Triangulation.....	9
2.3 MoSCoW	9
2.4 Interviews.....	10
2.5 Togaf 3 layer infrastructure drawing	10
2.6 Prototyping approach	11
3 Problem analysis.....	12
3.1 Preface	12
3.2 Used methods.....	12
3.3 Problem description	13
3.4 Prespectives	13
3.5 Conceptual model.....	16
3.6 Problem definition and main & sub-questions	17
4 Needs analysis	19
4.1 Preface	19
4.2 Used Methods.....	19
4.3 Stakeholder analysis.....	20
4.4 Requirements	22
5 Context analysis	25
5.1 Preface	25
5.2 Used methods.....	25
5.3 Current situation	26
5.4 Conclusion	27
6 Literature research	28
6.1 Preface	28
6.2 Used methods.....	28
6.3 Results	29

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS
applications implemented on a cloud-native platform.

7	Research conclusion.....	39
8	Design.....	40
8.1	Preface	40
8.2	Used methods	40
8.3	Cloud-native development environment comparison	41
8.4	Functional design	44
8.5	Technical design.....	47
9	Testing	49
9.1	Preface	49
9.2	Used methods	50
9.3	Tools and applications	51
9.4	Test environment and procedure	51
9.5	Test conclusion.....	53
10	Evaluation	55
10.1	Used methods.....	55
10.2	Research evaluation	55
10.3	Design evaluation.....	56
11	Advisory.....	57
11.1	Recommendations.....	58
12	Appendixes.....	59
12.1	Interviews.....	59
12.2	Architecture drawings.....	64
12.3	Closer look at the GitLab solution	70
12.4	Closer look at GitHub	73
12.5	Closer look at Azure DevOps.....	75
12.6	Closer look at lift and shift solution	76
12.7	Evaluations	77
12.8	Testing results.....	83
12.9	Guides and manuals.....	90
13	References.....	94
14	Table of Figures and tables	97

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

ABBREVIATIONS AND GLOSSARY

Word	Abbreviation	Definition
TACTICOS Combat management system	CMS	TACTICOS is a modular operating system for naval ships. It is used to as an application where radar data comes together with the fire systems.
Kubernetes orchestration		Orchestration is the automation of the management and coordination of applications, with the goal of improving efficiency and simplifying maintenance. Kubernetes is a tool that provides orchestration for containerized applications.
Integrated development environment	IDE	An IDE is (web)application that helps developers write code, an IDE offers plug-ins and other features that helps developers improve their code.
Artifact		Artifacts are bundles of Software applications, support documentation and informational content and more.
Container		A container is a package that includes all components needed to run an application. It is portable and provides a consistent environment for running applications,
Visual Studio Code	VSC	Visual Studio Code is an IDE application, an application for writing code.
Helm		Helm is a package manager for Kubernetes that simplifies the installation and management of applications in Kubernetes.
Cloud native Computing Foundation	CNCF	Is a foundation which focusses on making Cloud-native sustainable and new projects known.
Red Hat Enterprise Linux	RHEL	Red Hat is a Linux distribution mostly for commercial use, it offers extra support from Red Hat.
Development (security) operations	Dev(Sec)Ops	Dev(Sec)Ops is an abbreviation for the development process, where the security is a new add-on as security trends improved.

Table 3 Glossary and abbreviations

PREFACE

This thesis document is written for the assignment “Web-based development environment for TACTICOS applications implemented on a Cloud-native platform” and written by Max de Haas. The thesis is written as part of the graduation assignment for the bachelor degree in HBO-ICT in the IT Service Management direction. the study was conducted at Saxion Enschede.

The thesis consists of research, an advice, testing and results and a reflection.

The assignment is supervised by Herman Voortman from Saxion Hogeschool (Tutor) and Willy Boenink from Thales Hengelo (Company supervisor). I would like to thank them for their guidance, time and effort they put in to make sure I can fulfill the assignment. I would also thank every employee from Thales that helped me with interviews and Volonta for the fun outings. The assignment was educational and gave the opportunity to learn a lot from the Cloud-native world.

I hope that everybody will enjoy reading this Thesis and learn from doing so.

Max de Haas

1 INTRODUCTION

This is the Thesis assignment: “Web-based development environment for TACTICOS applications implemented on a cloud-native platform” This assignment has been given by Thales Nederland B.V. and will be carried out by Max de Haas.

Thales Nederland B.V. is the Dutch branch of Thales Group, Thales group has over 81000 employees and operates in over 44 countries worldwide. Thales Netherlands has three locations throughout the Netherlands with Hengelo (O) as its main location. In addition to Hengelo, there is a location in Huizen which deals with (radio) communication equipment, there is also a location in Delft which is a research and development location and (infrared) viewers and a location in Eindhoven which deals with cooling systems for military purposes. Thales Nederland has around 2250 employees.

The customers of Thales Nederland are mainly ministries (of defense) of countries spread around the world, the Dutch Railways and Boeing/Airbus.



Figure 1 Thales Nederland headquarters (GIB)

Thales is interested in moving their systems to Kubernetes clusters. Thales want to move their development environment to the cloud. They want to find out how they could improve on their current systems and either move them to a Kubernetes cluster or implement a new system to replace the current systems.

The main question that needs to be answered is:

How could a web-based development environment for TACTICOS applications be implemented on a cloud-native platform?

To find the answer of the main question a few sub questions must be answered first, these questions combined will provide an answer to the main question.

- 1. What is a web-based development environment?**
- 2. What is a Cloud-native environment?**
- 3. What is needed in development environment for TACTICOS applications?**
- 4. What are the requirements for a web-based development environment to be run on a cloud-native platform?**
- 5. How would the design and implementation of a web-based development environment be run on a cloud-native landscape?**

To make sure that these answers get answered the following method will be used: "Design Research" This process will start with the research phase consisting of the following research:

- Problem analysis
- Needs analysis
- Context analysis
- Literature research

After the research phase, the design phase will start. Build on the foundation in the research phase a design will be created in the design phase.

After the initial problem analysis, the stakeholders will be interviewed for the requirements in the needs document. After these requirements they will be sorted based on the MoSCoW method to prioritize the importance of the requirements. Once the requirements are clear, desk research will be used to further understand the concept of web-based development environments and cloud-native environment.

When the design has been designed it will be built in a Proof of Concept in the form of a prototype. The prototype will be tested and the results of the prototype, the design and the research will be used to draw a conclusion and give recommendation for Thales to use.

2 TECHNIQUES AND METHODOLOGIES

The research will be done using multiple different techniques and methodologies. In this chapter these techniques and methodologies will be described.

2.1 DESIGN RESEARCH

Design Research will be the main method that is used during the assignment. Design research is a customer-focused framework that uses multiple research methods to come to an end result.

2.2 TRIANGULATION

This technique is used to look at a problem from at least 3 different perspectives. This technique is used during the problem analysis and for this assignment the following perspectives will be used:

- Perspective of Thales
- Perspective of the user
- Perspective of the researcher

2.3 MOSCOW

MoSCoW is a technique that focuses on prioritizing the requirements that Thales has. It uses the capital letters to prioritize the needs in the following way:

Priority	Description
Must have (M)	The “Must have” requirements are for must have and is the letter used to mark the most important requirements.
Should have (S)	The “Should have” requirements are important to consider but not necessary.
Could have (C)	The “Could have” requirements are optional requirements that the customer desires. They can be implemented after the assignment/future research
Would have (W)	The “Would have” requirements are optional requirements that are not required for the assignment and for future research.

Table 4 MoSCoW table

2.4 INTERVIEWS

For the needs analysis interviews will be held, these interviews will be held with the stakeholders of the project. These stakeholders are identified in a stakeholder analysis. The goal of these interviews is to collect information on the needs and expectations of the design from the stakeholders. For the interview there will be a set of open questions prepared. These questions will be extended with follow-up questions during the interview depending on the answer the questioner gets.

The table below will show the stakeholders that were interviewed, their function and the goal of the interview.

Interviewee	Function	Goal
Willy	Infrastructure/software architect	Gaining a perspective of the problem, Gathering, requirements, context, conditions and other stakeholders.
Daniel	Cloud-native architect	Gathering, requirements, context and conditions for the assignment
Rob	Software architect CMS	Gathering, requirements, context and conditions for the assignment
Rick	IS/IT solutions architect	Gathering, requirements, context and conditions for the assignment

Table 5 Interviewees and goals

The interviews with the stakeholders can be found in the appendix “Interviews”.

2.5 TOGAF 3 LAYER INFRASTRUCTURE DRAWING

The TOGAF Standard is a proven best practice framework for enterprise architecture. This framework will be used to design and visualize the current architecture and a future architecture. These designs will be made in ArchiMate.

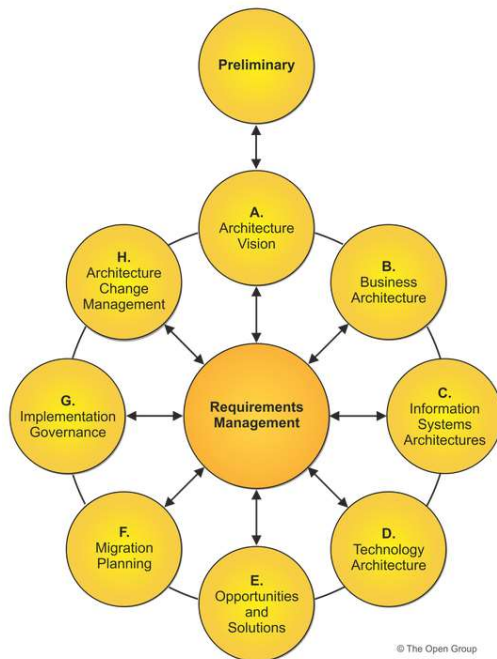


Figure 2 The TOGAF standard (The Open Group)

2.6 PROTOTYPING APPROACH

This method is used to develop the design by producing several smaller products, analyzing them with the key stakeholders, and testing them to conclude. There are three approaches for testing these are:

- The systematic approach – This approach uses a predefined plan with clear criteria to setup multiple prototypes.
- The deliberative approach – This approach uses communication with different people to setup multiple prototypes
- The prototyping approach – This method uses small end products to test and communicate them to the customer, the method can elements of both approaches mentioned before.

The guides and manuals can be found in the appendix "Guides & manuals". These guides contain all the steps taken and commandos used to setup the prototype used.

3 PROBLEM ANALYSIS

3.1 PREFACE

The problem analysis is the first step in the “design research” method used during the graduation period. The problem analysis will define the problem and make sure that the problem is correctly specified. The main and sub questions will be established. The problem analysis consists of the following chapters:

- Used methods – Describing all the methods used in the problem analysis
- Problem description – A description of the general problem during the research.
- Perspectives – A description of the different perspectives towards this problem.
- Problem definition – A summary of the problem constructed with the main question and sub-question

3.2 USED METHODS

This section describes the methods used during the problem analysis. The following methods are used in the problem analysis.

3.2.1 DESIGN RESEARCH

The problem analysis is the first analysis in the research phase according to the design research method. During the problem analysis it is the goal to use different methods to create a clear problem. The outcome of the problem analysis will help in with future research.

3.2.2 TRIANGULATION

The goal of this method is to see the problem from various perspectives. Different perspectives will be used to look at a cause or (part of) the problem. Triangulation needs a minimum of 3 perspectives to get good results and all the perspectives together should enlighten the problem. Interviews will help gather information about other perspectives.

3.2.3 INTERVIEW

The goal of the interviews is to gather information. The first step is an interview with Willy (Supervisor from Thales). The goal of this interview is to gather information about the problem and to create a list of stakeholders. The step after is to interview the stakeholders from the list to gather their perspective, view on the problem, their needs and demands. Their perspectives will be part of the above-mentioned triangulation.

3.3 PROBLEM DESCRIPTION

Thales is always interested in future solutions and technologies they might need in the future. Cloud-native solutions are becoming more important every day and more and more people are becoming dependent on them. Much research has been done on the security and deployment of containers. As the trend continues and security improves, more companies are deploying containers to run applications.

For this assignment, containers play a big role. Thales might want to build their development environment on a Cloud-native environment. Before Thales can implement a system like that, they need to know what problems can occur. More and more applications are compatible in a Cloud-native environment and more applications get updated to get rid of the above-mentioned problems.

This research must come with a clear view of the current Cloud-native solutions and how certain applications will work on a Cloud-native environment, what are the pro's and what are the cons and which solutions for each part in the development area is the best choice for Thales to offer to the Naval Domain according to requirements and conditions. Everything must be well documented so the results can be reproduced for future usage.

3.4 PRESPECTIVES

Perspectives will be used to clarify the problem. these perspectives were chosen in consultation with the problem owner (Willy). The perspectives chosen are from various functions within Thales that may be affected by a new development environment. The perspectives chosen will be from Thales, Solutions architect, software architect, (cloud)infrastructure architect and literature.

3.4.1 PERSPECTIVE THALES NEDERLAND B.V.

Thales is always interested in researching future solutions to improve on security, performance, and functionality. The goal of these research project is to have information about the latest techniques on the market, the research done will help Thales see how and whether a new technology will improve any aspect of the current system. Thales want to improve their systems and products but does not want to spend too much money.

3.4.2 PERSPECTIVE SOLUTIONS ARCHITECT

The function of a solutions architect within Thales is to set up and maintain the systems of Thales. Thales is limited in the type of solutions they can use especially cloud solutions. Thales is not allowed to use a public cloud solution, so any solution must be hosted in their own environment. The EAST team has the knowledge to maintain a containerized

network but has not yet attempted to move to a full containerized development environment.

3.4.3 PERSPECTIVE SOFTWARE ARCHITECT

Software developers will have to deal with a big change in their IDE (Integrated development environment.) The reason is that in the current environment, the IDE is the only application running locally, while the other applications can be run locally, but most employees already run them in the Web browser.

The research should confirm if a web IDE will be comparable to the physical application. The git, building and artifact tools are already accessed by web so any changes there would probably be on the back end and that is something not relevant for the software developers.

3.4.4 PERSPECTIVE (CLOUD)INFRASTRUCTURE ARCHITECT

Thales is always looking at improvements. If they were to implement a Cloud-native environment and start running more applications in container clusters then possible problems might come forward. Infrastructure architects help with finding these solutions. In this case Architectures are mostly interested in the advantages, disadvantages, and the performance of a cloud-based development environment.

Thales must build everything in their own private cloud, so public cloud solutions are not an option. The infrastructure architect does look positive for the benefit this research can bring.

3.4.5 PERSPECTIVE USERS

Thales has a lot of developers, a problem with that is that there are a lot of different programming languages and applications used by the users. The problem that can come forward from here is that one IDE can work while another one cannot work. Also, the IDE is the most important part for the developer. Developers do not need to know how certain applications work; they just want it to work.

3.4.6 PERSPECTIVE LITERATURE

The trend is changing since 2010, moving faster towards cloud-native solutions. The assignment is focused on the container and the DevOps side.













	Development Process	Application Architecture	Deployment & Packaging	Application Infrastructure
~ 1980	Waterfall 	Monolithic 	Physical Server 	Datacenter 
~ 1990				
~ 2000	Agile 	N-Tier 	Virtual Servers 	Hosted 
~ 2010	DevOps 	Microservices 	Containers 	Cloud 

Figure 3 Cloud-native trend (Oracle)

A cloud-native environment can provide several important benefits over an older environment. These benefits are:

- **Independence** : Cloud-native application can be run and maintained independently
- **Resilience** : A well-designed cloud-native application can stay online even if the infrastructure fails.
- **Based on standards** : Cloud-native services are built with opensource standards in mind and lower the dependency of certain providers.
- **Flexibility** : A Cloud-native environment can be setup in different ways and can be managed in different ways.
- **Automation** : There are multiple ways in why automation is benefit. There is auto scaling and the way containers communicate using API's
- **Zero downtime** : A well build container orchestrator has built in options to perform rolling updates also when a pod stops working a new pod will be built to replace the broken pod.

3.5 CONCEPTUAL MODEL

The goal of the conceptual model is to visualize the benefits and the goals of the main features of a container solution. The results and some of the information come from the literature research (chapter 6 in this document). A container solution has 4 main features at the time of this project. These features bring benefits and these benefits are used to fulfill the goals.

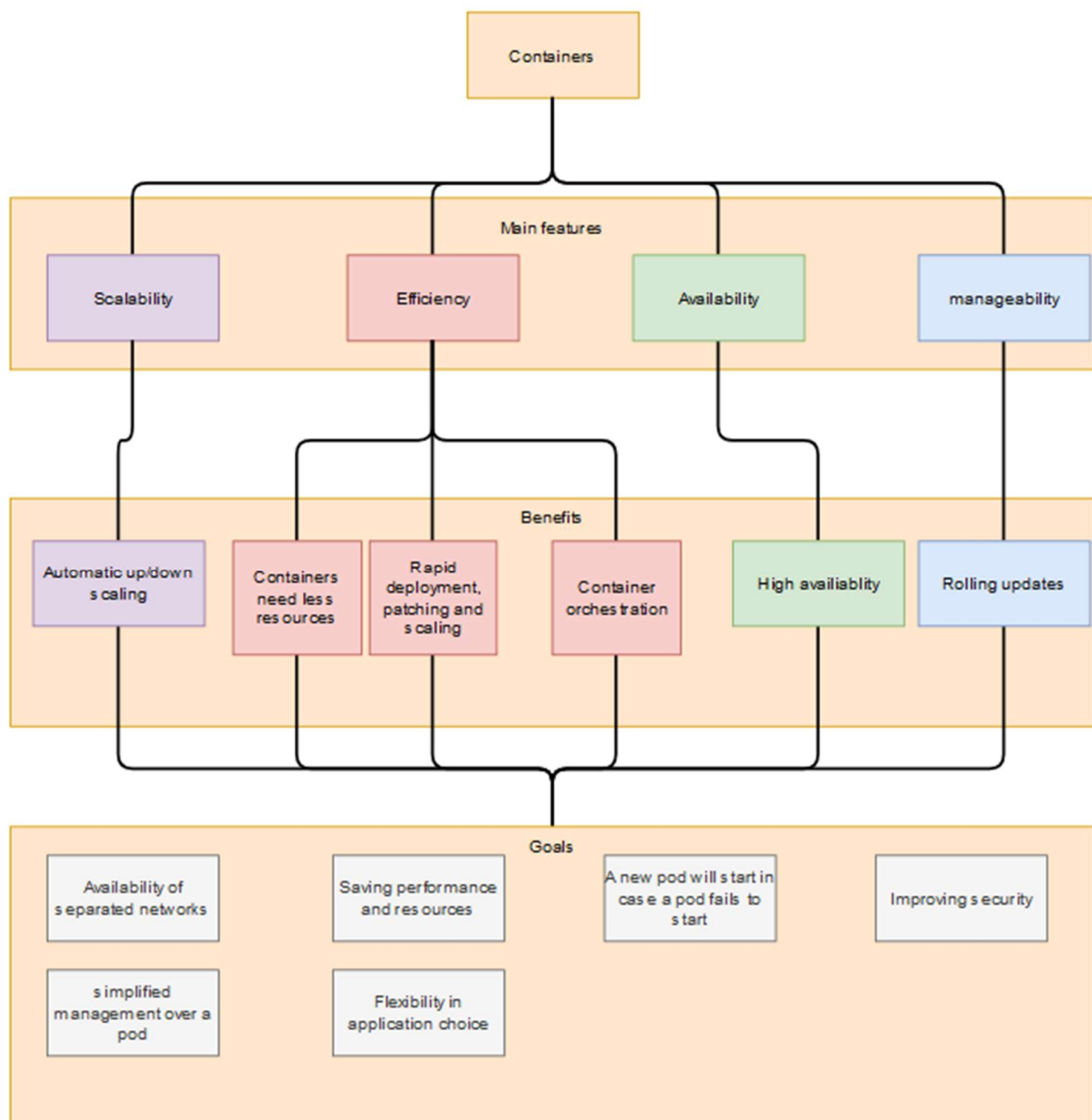


Figure 4 Conceptual model

3.6 PROBLEM DEFINITION AND MAIN & SUB-QUESTIONS

The problem definition defines the problem using pre-defined questions that are created by using the perspectives and consultation with the problem owners. There is one main question that is answered once all the sub-questions have been answered. The main question itself defines the problem that will be researched.

The main goal for Thales for this assignment is to gain information and a vision on how they could build a development environment in a Cloud-native environment, Thales is interested in the solutions, the advantages and consequences that might come with it. The following main question is created for this assignment:

“How could a web-based development environment for TACTICOS applications be implemented on a cloud-native platform?”

The following sub questions are used to answer the main question:

- 1) What is a web-based development environment?
- 2) What is a Cloud-native environment?
- 3) What is needed in development environment for TACTICOS applications?
- 4) What are the advantages and disadvantages of going Cloud-native?
- 5) What are the requirements for a web-based development environment to be run on a cloud-native platform?
- 6) What web-development tools are available on the market?
- 7) How would the design and implementation of a web-based development environment be run on a cloud-native landscape?

A description of why a sub question was chosen and where in this study the sub question will be addressed can be found in the following table:

Sub question	Description	Covered in
1	It is important to understand the basics of web-based development and how it differs from normal developing.	Literature study Needs analysis Context analysis
2	Containers are important part of a Cloud-native environment, which is why it is important to understand and figure out the pros and cons of a Cloud-native environment.	Literature study Needs analysis Context analysis
3	To design a new development environment or improve the current one, it is important to understand what is needed for a development environment.	Needs analysis context analysis literature study design

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

4	Knowing the advantages and disadvantages of the Cloud-native platform helps in designing a new environment as the strong and weak points can be determined.	Needs analysis context analysis literature study
5	The requirements that are needed come from 2 sides, the view of what Thales needs and the requirements an application has. IT is important to know these requirements to figure out what resources are needed.	Needs analysis literature study
6	For this question it is important to know what are the main programming languages used (for TACTICOS), This way the best applications for these languages can be researched and tested.	Needs analysis design prototyping
7	A new Cloud-native design would mean that the underlying infrastructure must be changed. It needs to be researched what the effect will be of a change like this.	Context analysis Needs analysis design prototyping

Table 6 Sub question descriptions and coverage

4 NEEDS ANALYSIS

4.1 PREFACE

The second step of the “Design Research” method is the needs analysis. The needs analysis is used to determine the stakeholders using a stakeholder's analysis. The stakeholders will also be used to gather their needs and requirements. After the list of requirements has been drafted up, it will be sent to the stakeholders for an extra review. The outcome of this analysis is the checked list of requirements, this list is important for decisions made in the future and the outcome of the assignment. The following chapters will be in the need's analysis:

- Used methods - Describing all the methods used in the Needs analysis.
- Stakeholder analysis – Analysis to determine all stakeholders for this assignment.
- Requirements – A list of requirements and needs are setup and prioritized. The list contains all the relevant requirements and needs from the stakeholders and verified with the boundaries of the assignment.

During the needs analysis 1 sub question will be answered this is the following question “What are the requirements for a web-based development environment to be run on a cloud-native platform?”, it will be answered All other sub questions will have extra information.

4.2 USED METHODS

4.2.1 STAKEHOLDER ANALYSIS

To determine and analyze the stakeholders in this project and their role inside Thales. To analyze the stakeholders a 3-step plan will be used. The first step is to determine the stakeholders, the second step is to sort the stakeholders and the third step categorize the stakeholders. The graduation supervisor will start by giving some stakeholders and more have been added during interviews with the recommended stakeholders.

The next step is to prioritize the stakeholders in their power over this project and their interest in this project. What stakeholders have high interest in the project and would like to be updated and what stakeholders have high power over the project and can stop it if they want.

4.2.2 MOSCOW

MoSCoW will be used to prioritize the requirements. This method will prioritize the requirements using the following words: Must, Should, Could and Would. Where Must are the requirements that are required and would are the requirements that are not necessary but nice to have.

4.2.3 INTERVIEW

An important part of the needs analysis is to gather requirements from stakeholders, the method used will be the interview. Interviews will help to know more about the stakeholder's perspectives, their views, wishes, demands and their requirements.

4.3 STAKEHOLDER ANALYSIS

4.3.1 STEP 1: IDENTIFYING STAKEHOLDERS

To find the right stakeholders for this assignment, an interview was scheduled with Willy Boenink. During this interview, different perspectives, departments, and stakeholders were discussed and from this the stakeholders were identified. In addition, additional stakeholders were added during the stakeholder interviews.

Stakeholder	Function
Willy Boenink	Software architect
Rick Jansen	IS/IT solutions architect
Daniel van Gils	Cloud-native architect/Systems architect
Rob ten Hove	Software architect CMS
Lex Emmens	Software engineer

Table 7 Stakeholders and their function

4.3.2 STEP 2: SORT STAKEHOLDERS

The second step is to sort the stakeholders in Primary stakeholders (directly involved) or secondary stakeholders (Interested parties), this sorting method is used to reveal their involvement in the assignment. There are no external stakeholders in this project.

Primary stakeholders (Directly involved)	Secondary stakeholders (Interested parties)
Willy Boenink	Rick Jansen
	Daniel van Gils
	Rob ten Hove
	Lex Emmens

Table 8 Sort primary/secondary stakeholder

4.3.3 STEP 3: CATEGORIZE THE STAKEHOLDERS

The next step is to categorize the stakeholders, it is important to understand their power and their interest in the assignment. There are 4 main categories for a stakeholder. (See stakeholder categorization matrix below) The categories help define the importance of the requirements and possible exceptions from certain requirements.



Figure 5 power/interest grid example

To prioritize the stakeholders a Power/Interest Grid will be used, this grid consists of 4 categories.

- **Manage closely** - Stakeholders from this category require a lot of effort to keep up to date and satisfied.
- **Keep satisfied** - Stakeholders from this category require some status updates during the assignment to know what the status is of the assignment.
- **Keep informed** - Stakeholders from this category require regular updates and information about the assignment pure out of interest.
- **Monitor** - these stakeholders have low power and low interest but are involved in a way.

Stakeholders are also color coded in the Power/Interest grid; the colors have to the following meaning.

Red: stakeholders being critics and are the most important and require regular updates.

Orange: stakeholders are neutral they know about the project and require a result in the end.

Green: are supporters and are important during the project and like the project but also do not require special needs.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

Stakeholders	Influence	Power	Color	Categorization
Willy	High	High	Red	Manage closely
Rick	Moderate	High	Orange	Keep satisfied
Daniel	High	Moderate	Green	Keep informed
Rob	Low	Moderate	Orange	Monitor
Lex	Low	Low	Green	Monitor

Table 9 Stakeholder's categorization

The Power/Interest grid can be finished with the information from the table above.



Figure 6 Power/Interest Grid

4.4 REQUIREMENTS

The requirements are a key element in determining the result. The requirements are gathered using interviews. Interviewing the relevant stakeholders is the main method used to gather information which can turn into requirements. The requirements will be prioritized using the MoSCoW methodology. The requirements will be prioritized in consultation with Willy. Requirements fall in 3 categories, Business-, User- and system-requirements. The final requirements are controlled and validated with the main stakeholder and Thales supervisor Willy.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

	ID	Description	MoSCoW	Functional	Non-functional	Gained from
Business requirements	BR-01	A possible container solution should not be tailor made and be commercially available.	Must		X	Daniel, Rick
	BR-02	A design should be able independent. One container solution/program can be replaced with another.	Must		X	Rick, Daniel
	BR-03	The design must be well documented.	Must		X	Willy, Rick
	BR-04	Employees of these teams must be able to reproduce the proof of concept.	Must		X	Willy, Rick
	BR-05	The design should not have a vendor lock-in	Must		X	Rick, Willy, Daniel

Table 10 Business requirements

	ID	Description	MoSCoW	Functional	Non-functional	Gained from
User requirements	UR-01	The IDE must be accessible from a browser.	Must	X		Willy
	UR-02	Developers must have access to the following functions in an IDE: <ul style="list-style-type: none"> • Compiler • Debugging tools • Code completion • Refactor 	Must	X		Rick, Rob
	UR-03	Developers have access to plugins for testing	Could	X		Rob
	UR-04	Artifacts must be built automatically	Must		X	Willy, Rick, Daniel, Rob
	UR-05	Developers must have access to a version control and merge support.	Must		X	Willy, Rick, Daniel, Rob

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

	UR-06	Developers must have access to an artifact storage.	Must	X		Willy, Rick, Daniel, Rob,
	UR-07	The design must support a static code check for example SonarQube and Coverity.	should		X	Rob
	UR-08	The advantages and disadvantages of possible solutions must be described	Must		X	Willy
	UR-09	The features of different solutions must be described and documented	Must		X	Willy

Table 11 user requirements

	ID	Description	MoSCoW	Functional	Non-functional	Gained from
System requirements	SR-01	The finished product must run in a local environment with no access to the internet and be managed locally.	Must	X		Willy, Rick, Daniel, Rob
	SR-02	The system must have high availability.	Should	X		Willy, Rick, Daniel
	SR-03	The system should have scalability for at least 500 engineers.	Should		X	Willy, Rick, Daniel
	SR-04	The new environment must have auto scaling capabilities.	Could	X		Willy, Rick, Daniel

Table 12 System requirements

5 CONTEXT ANALYSIS

5.1 PREFACE

The Context analysis is the third step in a “Design research” method. This analysis is used to determine the environment in which the intervention will take place. When the current situation is clear, the second step is to define the conditions for the solution. This is important for the progress of the task and the result. The result of this analysis is an overview of the current situation in which the intervention will take place and a list of intervention conditions. The following sections constitute context analysis:

- Used methods – Describing all the methods used in the context analysis
- Current Situation – A drawing and explanation about the current situation What the solution is designed for.
- Conclusion – Overall conclusion of the context analysis and a list of conditions.

Some points of the sub-questions are mentioned or provide information in this chapter these are the following sub-questions:

1. What is a web-based development environment?
2. What is a Cloud-native environment?
3. What is needed in development environment for TACTICOS applications?
4. How would the design and implementation of a web-based development environment be run on a cloud-native landscape?

The goal is that these questions will help define possible conditions for the assignment. The conditions will be used to find a design that fits Thales their needs and wishes the best. The previously defined requirements and conditions will be used to clarify the intervention. At the end of the chapter the question “What are the conditions for the solution?” from the “Design research” method will be answered.

5.2 USED METHODS

5.2.1 INTERVIEW

Interviews are an important part of the whole assignment. Interviews will be used to gather information about the conditions and a view of the current situation of the environment. The interviews will be taken with the stakeholders, the views of the stakeholders will help with defining the problem, the assignment and finding a solution.

5.2.2 ARCHITECTURAL MODEL

The goal of architectural models is to visualize the infrastructure and processes of a company. The architectural model will be used in the context analysis in the form of a drawing of the current situation. This drawing will be made in ArchiMate using the “TOGAF ADM” methodology. The drawing delivers input

5.3 CURRENT SITUATION

The information gathered on the current situation comes from interviews and information from Thales their wiki. These interviews can be found in the Appendix "Interviews". The information gathered from the interviews can help in designing an architectural model of the current situation. This model will also be the base and comparison for a possible solution.

Thales does not use a container solution in the current development environment as the current solution is virtualized environment using virtual machines to host the applications that are important for the development environment. The reason Thales is making the step towards a Cloud-native solution is that the environment can be more efficient in resources, easier to manage and etcetera. Besides the above-mentioned advantages it is also important to note that vendor lock-in will not be as impactful, through the advantage of a Cloud-native standard, each application runs independently, this gives the rolling updates so individual containers can get their update if they are not used at that time.

Thales has always been looking into new technologies, while container orchestration is already used inside different parts of the company it has never been used to manage their development environment. Container orchestration tools like Kubernetes and VMware Tanzu are widely used by big companies like Spotify, Adidas and many more. A cloud orchestration tool offers great solution in managing applications.

Any new Cloud-native solution will have the most impact on the systems. The servers will have to use different software (for example Kubernetes). In return the programs that will run on the new software will receive a better manageability, more efficient with resources and better auto scaling to support the on demand needs of the software. Deploying a Cloud-native environment could help Thales If the chosen and tested solution meets the requirements that Thales said it is up to Thales to decide if a new solution will be implemented at the end of the assignment.

In the following chapter an architectural model is made using ArchiMate, this model shows the current infrastructure on which a new solution will be designed.

5.3.1 ENTERPRISE ARCHITECTURE MODEL

The models can be found in appendix "12.2.1 Current situation"

The model is based on the answers gathered during the interviews. The drawing is split up in a developer environment and the maintain/manage side of the environment. This has been chosen as these maintainers and the developers will receive the biggest change.

In the development environment there are 3 actors that are important. The developer their job is to write the code, a test engineer to test the code and make sure it is as safe as possible to use and a release manager that makes sure the artifacts are working as intended and decides when to release an artifact. The issue management is a part that is out of the scope for the assignment but there the assignments a development process, When the code is written in the IDE and updated/uploaded to the version control system. If the code is ready to be built it will be built and tested in Jenkins, if the code is tested and build then the artifact will be (bundled) released and put in an artifactory. All these systems/ applications are hosted on specific application servers. The development environment itself is part of a Citrix environment and the applications within are hosted on a server where the different physical servers are controlled VMWare vSphere.

On the maintainer/manage side there is one actor (There are specific teams for it). Their job is to maintain, update and configure the environment so that it keeps working as intended. They have access to all the management/admin pages and applications and maintain/control the environment trying to make sure that there will be the least amount of downtime necessary.

5.4 CONCLUSION

During the context analysis and the interviews done some conditions became visible. These conditions will be used in designing a prototype, and are important for after as the company has to adhere to these conditions.

The following conditions have been established:

- Must work within a Cloud-native Environment.
- Solution must work within a Linux based operating system.
- Preferred solution with a Kubernetes orchestrator.
- Applications in a solution must be able to work together while running stand alone.
- Solution must have an ease of management.
- The system should be able to run in a production environment.
- Must integrate with different applications to improve further.

6 LITERATURE RESEARCH

6.1 PREFACE

The literature study is the fourth step in a “Design research” method. The goal of the literature study is to write down the latest information and research and theories on the sub questions of the assignment. The literature study is an important step as the main result of the literature study will be the design principles. The literature study consists of the following chapters:

- Used methods – Describing all the methods used in the Needs analysis.
- Results – Certain sub questions will be researched and their answers will be written down.
- Research conclusion – The conclusion of the literature study will be written down.

The goal of the literature study is to help give valuable information to the sub question; the following sub questions will be addressed:

- What is a web-based development environment?
- What is a Cloud-native environment?
- What are the advantages of going Cloud-native?
- What web-development tools are available on the market?
- How would the design and implementation of a web-based development environment be run on a Cloud-Native landscape?

6.2 USED METHODS

6.2.1 DESK RESEARCH

Desk research is a research method where the researcher finds reliable sources while sitting behind the desk, information can come from all types of sources so it is the researches' goal to see how reliable the sources are and what information can be used from a source.

6.2.2 SNOWBALL METHODOLOGY

The snowball methodology is another research method that is used during the literature study. The snowball method is used to dive further in to certain subjects, when something interesting is found the researcher will read further in that subject. The more in dept research method makes sure that the sources are more reliable and that there are multiple sources share the same answer on a subject. This way Snowball realizes a Literature Study that contains deep information on topics from multiple sources. (Haveman, Hageraats, & van der Linden, 2016)

6.3 RESULTS

6.3.1 LITERATURE QUESTION 1: WHAT IS WEB BASED DEVELOPMENT?

Web based development is a new and upcoming technique that focusses on moving the development environment to the cloud. A well-designed development environment consists of the following tools: An IDE, version control system, building tool/pipeline and an artifactory. First the tools/functions must be explained.

An IDE (Integrated Development Environment) is a software application that provides a great number of features for a developer to write code. A few of these features are build automation, compilers and debuggers used to control, write and change written software code.

A version control system is a system that tracks and manages software code. This tool is useful as it can track every change made to the software code and it allows the developers to compare earlier versions of the code if a mistake has been found. This tool is especially useful when there is a development team working on the same code, every developer can see what other developers did and control their work. A developer can also pull the latest version of the code from the version control system.

Software building tools/pipelines are used to build the written source code and create a software artifact that can be run on a computer. The build tool that supports the building of the code, these features can be automation, testing, extra version control and compilation. The build is also called an artifact. The artifact will be stored in an artifactory.

The artifactory is a repository where all the builds will be stored and managed. An artifactory makes sure that developers can easily use code and builds from other teams or previous projects and their dependencies. After the source code has been build and stored in the artifactory the software packages will be ready to be used. These software packages are called artifacts

For a web-based development environment all the above-mentioned programs/tools must be usable in a web browser. The development environment and specifications are different for all companies but these tools/functions are the necessities.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

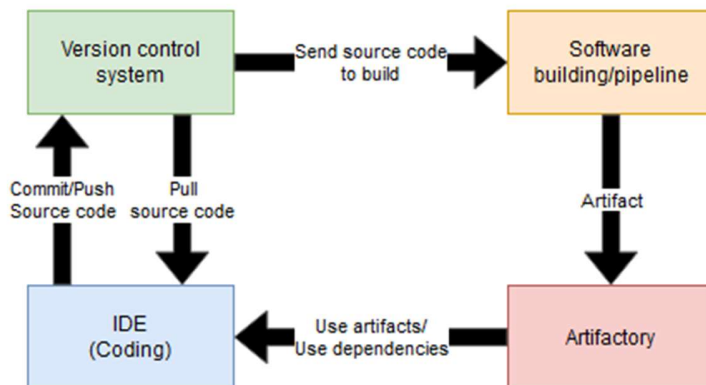


Figure 7 Development environment drawn

CONCLUSION

In short, a web-based development environment is the same as a standard development environment but with web-based possibilities and access from a web browser. The current environment supports web-based development but is not build on a Cloud-native layer.

6.3.2 LITERATURE QUESTION 2: WHAT IS A CLOUD-NATIVE ENVIRONEMNT?

Cloud-native has been up and coming for a few years with large and small companies moving towards a Cloud-native platform. With new technologies coming every day, the Cloud-native environment is an interesting choice for tech industries. To help improve the Cloud-native environment the CNCF (Cloud-native Computing Foundation) has been created by the Linux foundation in 2015, with the following goal "The Cloud-native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible for everyone." (Cloud-native Computing Foundation (CNCf), 2022) The CNCF helped redefining how software has been built and made a collective landscape of all Cloud-native apps.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

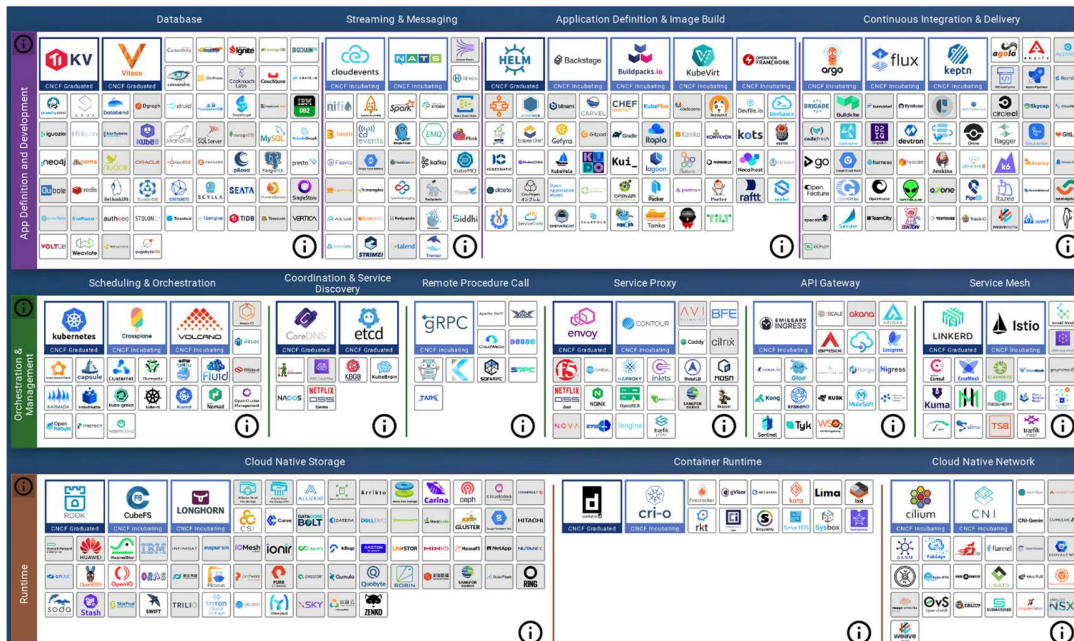


Figure 8 Cloud-native landscape (CNCF)

The CNCF gives the following explanation for Cloud-native “Cloud-native technologies empower organizations to build and run scalable applications in different environments such as public/private/hybrid clouds, with technologies such as: containers, service meshes, microservices, immutable infrastructure and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.” (Cloud-native Computing Foundation (CNCF), 2022)

The CNCF is hosting projects and has successfully build successful projects together with the support of over 150.000 contributors and big companies like Apple, Google, Amazon, Microsoft and many more.

The Cloud-native environment consists of 2 main building blocks and 2 trends, the building blocks are containers and microservices and the 2 trends are Cloud and DevOps.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.













	Development Process	Application Architecture	Deployment & Packaging	Application Infrastructure
~ 1980 ~ 1990	Waterfall 	Monolithic 	Physical Server 	Datacenter 
~ 2000	Agile 	N-Tier 	Virtual Servers 	Hosted 
~ 2010	DevOps 	Microservices 	Containers 	Cloud 

Figure 9 Cloud-native trend (Oracle)

Containers are lightweight virtual machines, due to their lightweight they are efficient, highly portable and scalable. Containers contain only the application libraries and processes they need. Containers are isolated from each other, this way multiple containers can run on 1 server without impacting each other, containers are also faster in deployment time and have better security if configured right. All these points together make containers a lightweight alternative to virtual machines and physical servers. Multiple containers can be managed using an orchestrator, an orchestrator has many features such as autoscaling, monitoring, balancing and repairing containers.

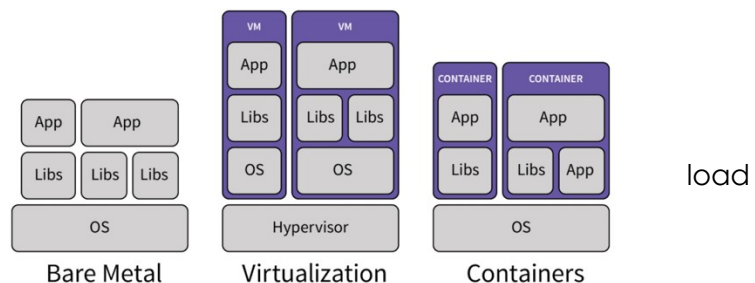


Figure 10 Containers

The other building block is microservices. If you have an application, it is most likely being made by 1 company with multiple functions. Microservices are mini applications where you can combine as many microservices as you want to create the same or better result as an actual application. With these microservices an company can configure the exact functionality they want all from different vendors which all use the Cloud-native standards and are easy to change. 2 of the biggest benefits are Rapid innovation and greater levels of data center automation. What is meant with rapid innovation is that a company do not have to

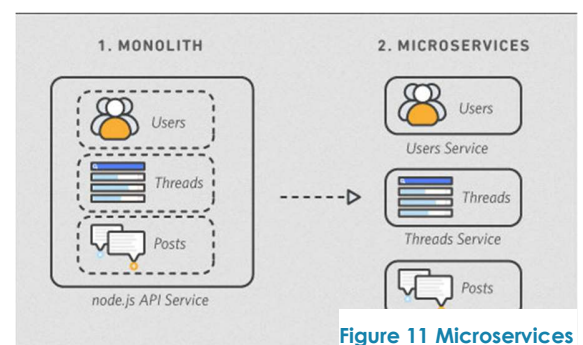


Figure 11 Microservices

wait till their vendor updated the application to have the feature they desire, there is a large support of open source content to create microservices. The second point about automation, microservices connect through scripted processes such as API's, these API's can be used to automate most systems. In the picture at the start of the next page a preview off some microservices is shown.

The 2 trends that gained a big foothold is DevOps and Cloud are more known. The goal of the Cloud is to make applications available to anyone and everywhere with the only necessity being an internet browser and internet.

DevOps is the CI/CD pipeline companies start to use to streamline their software creation process and make it more efficient. The DevOps circle consists of multiple of all the 8 steps in a development process.

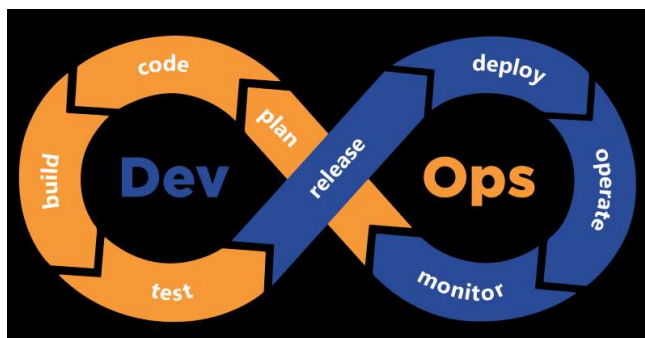


Figure 12 DevOps drawing

CONCLUSION

Cloud-native means that applications and systems are built with the advantages of a Cloud delivery system. Some examples on what is needed to implement Cloud-native systems are the use of micro-services, containers, the cloud itself and the use of DevOps structures. The goal of Cloud-native architecture is to create more on-demand systems.

Thales is using some Cloud-native environments within their company. Within the Development environment Thales only has DevOps. They have indicated that they are interested in setting the first steps towards containers for parts of the development environment.

6.3.3 LITERATURE QUESTION 3: WHAT ARE THE ADVANTAGES AND DISADVANTAGES OF GOING CLOUD-NATIVE?

The Cloud-native trend made a big leap forward in 2015 when the CNCF has been founded and announced Kubernetes 1.0, Kubernetes is an open-source container cluster manager. A big advantage of the Cloud-native environment is that mayor companies like Google, Red Hat and VMware pledge their ideas, knowledge and employees to support the Cloud-native community and open-source projects within

the CNCF. So, the strength of most Cloud-native applications is that they are open source and/or backed by major world leading companies.

To start with the advantages of a well build Cloud-native system.

- Reduced time to market
 - This means that due to the efficiency of Cloud-native applications that a development process can be quicker and an application can be delivered earlier.
- Ease of management
 - Cloud-native solutions help bring more monitoring tools and more options for managers to setup systems.
- Reduced cost
 - Efficient applications help reduce cost as they use less resources, auto scaling help scale the number of applications/instances running by the demand. If there is no demand only 1 or 2 instances of that application will be running. Some Cloud-native options are open source that can help reduce licensing cost.
- Reliable systems and reduced downtime
 - Most Cloud-native applications have advanced monitoring and logging tools which can help identify errors before they happen
- No vendor Lock-in
 - In the Cloud-native landscape, a company has the option to take a different application for all the different systems. A company is not bound to one vendor delivering the system without options to integrate with other vendors.
- Scalability and flexibility
 - As mentioned with the cost part, an environment is auto scalable that manages to spend the resources on the right instances. It can also help with setting up a specific environment in a faster timeframe than a virtual machine.

The Cloud-native environment has one major downside which is the loss of portability. This can be a problem for applications being built for 1 environment and must be ported to a different environment which can take a long time rewriting and refactoring code. If an application is built using a certain set of programs and the code needs access to a certain artifact that is stored in an artifactory. If the whole system is changing and the artifactory is changed it could mean that that part of the code must be rewritten.

For a company it can be a choice if they want to future proof their environment and invest the upfront cost and manpower to refactor the code or keep the artifactory as is and change the systems around it. A company must look at the advantages of going Cloud-native one a case by case and application by application basis.

CONCLUSION

The support for Cloud-native systems and applications is great as it is backed by multiple industry leading companies. It also brings more benefits than disadvantages for the companies that switch towards Cloud-native. The biggest advantage is the efficiency as it will improve the resources used on the server. The biggest disadvantage is that the company can lose some portability in their applications.

This change might not be a problem for Thales as they have a large amount of developers and have the option to set apart a team to start working on it. A solution might come that will make porting less of a disadvantage.

6.3.4 LITERATURE QUESTION 4: WHAT WEB-DEVELOPMENT TOOLS ARE AVAILABLE ON THE MARKET?

It is necessary to look at the first literature question to understand what is needed for web development. In the first question the following 4 tools were discussed: IDE, Version control system, Software building/Pipeline and an Artifactory.

There are multiple ways to setup a complete development environment, here are some examples:

- **All in one solution** – 1 application for the whole development environment.
- **Lift and shift** – Lift the current applications and deploy them on a Kubernetes cluster instead of a virtual machine.
- **Individual Cloud-native applications** – Chose an option to fulfill 1 (or more) of the 4-parts of a development environment.
- **All in one with support of other Cloud-native applications** – All in one solution might not fulfill the full package of requirements and can be supported by different applications.

All in one solution are products that have at minimum the features needed to replace the current development environment. In short, the application must have a build in IDE, a version control system, code building and an artifactory. Most big Cloud providers like AWS, GCP and Azure have built their own all in one system that can be used on their own platform. Other well-known products are GitHub and GitLab, both are more used by the coding community and support the 4 most important parts of a development environment.

The lift and shift method are a common way to move an environment to the Cloud, the meaning behind lift and shift is that the application will be lifted off from the current server and put in the Cloud. For the assignment the main goal is to move towards a Cloud-native environment the to account for this goal a small change must be made, instead of moving it the Cloud it must be moved to a container. The container can be run in an application like Kubernetes (K8S).

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

The third option is to use a Cloud-native application for each part of the development environment. Using the Cloud-native applications, it should be easy to couple and decouple different applications and processes to each other using API's, this way a new full Cloud-native environment can be setup.

The fourth option the all in one with support of other Cloud-native applications, as explained before an all-in-one solution might not fulfill all the requirements a company has, so a company can decide if they want to add an extra

To start with the Cloud-native IDEs, Cloud-native IDEs is a new and emerging technology with the 2 most widely used desktop IDEs (JetBrains IntelliJ, Microsoft Visual Studio Code) releasing their product in the last 2 years.

The second part of a development environment is about version control. While there are a lot of different vendors for a version control system, the most known/used programmers all use the GIT open-source standard. GIT is an open-source distributed version control system that uses branches to control a version. The developers work in a developing branch and if needed a push request can be done to make changes in the master branch.

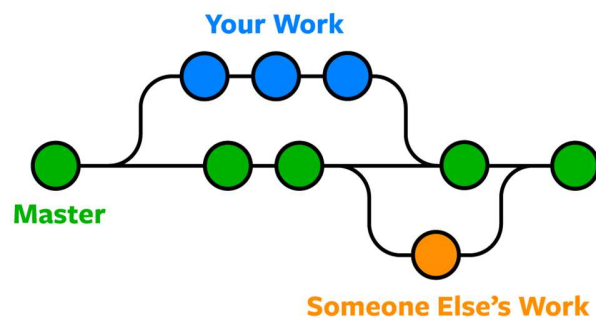


Figure 13 What is GIT

Some alternatives for Git solutions are Azure DevOps Server and Helix Core.

Building code is the next step in the process. Build code has been tested and is ready to be used as an application or can be used as a dependency in other code. The reason to build you code is to make an application from the code that is written. Thales is currently using Jenkins for it; Jenkins is one of the most known and oldest build automation tools for a CI/CD pipeline. Many companies joined the CI/CD trend with their own application as shown in the screenshot below.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

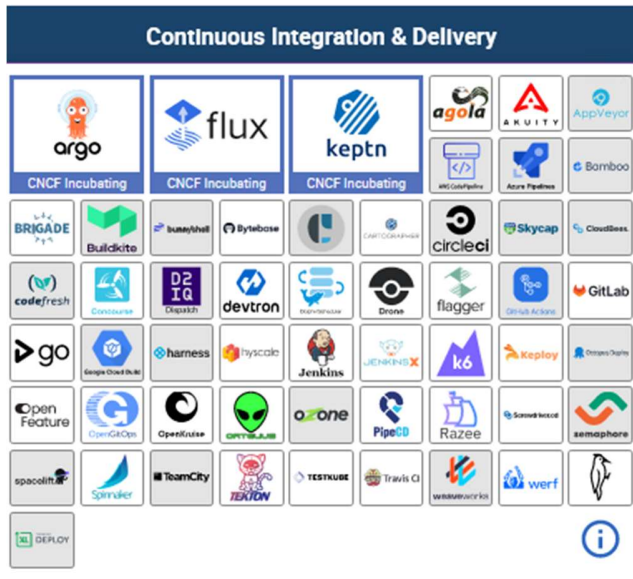


Figure 14 Cloud-native computing foundation CI/CD

Not all the applications shown are direct competitors to Jenkins but some great alternatives are Circle CI, GitLab and Bamboo.

The final step is storing the build artifacts, hereby is automation key in the process from start to finish, so it is good to know that the build Pipeline can integrate with the artifactory and the artifactory must integrate with the IDE. Cloud-native Artifactory's are uncommon and most require a container orchestrator like Kubernetes to function. Some Cloud-native solutions are GitHub and GitLab and some solutions that need Kubernetes to function are JFrog and Nexus.

CONCLUSION

In short, there are 4 ways to come to a solution using field proven methods. These are:

- **All in one solution**
- **Lift and shift**
- **Individual Cloud-native applications**
- **All in one with support of other Cloud-native applications**

The goal is to use these methods to help design a solution that fit the requirements. Some applications have been mentioned to redesign the current system or improve the current system to a Cloud-native environment. A few designs and further research into all the mentioned products will be worked out in chapter 8 "Design".

Inside of Thales there are teams that have experience with migrating systems, the knowledge is there and so is the experience.

6.3.5 LITERATURE QUESTION 5: HOW WOULD THE DESIGN AND IMPLEMENTATION OF A WEB-BASED DEVELOPMENT ENVIRONMENT BE RUN ON A CLOUD-NATIVE LANDSCAPE?

If a company were to switch to a Cloud-native environment, then the company must take a set of steps to overcome the switch. While the set of steps to take are mostly the same with the exception the lift and shift concept. The main problem a company would face is porting their code for a new solution, any code that currently or has been written and that uses artifacts must be temporarily stopped, this is due to the location of the dependent artifacts changing to a different artifactory.

While this process can take a very long time and would bring a high cost in salaries of developers who are not able to perform certain tasks as they must change the code. The lost money can be earned back during the upcoming years as the efficiency improvement can reduce the development time.

The moment that code from older projects has been ported to the new artifactory and that new projects have started using the new artifactory. The rest of the systems can be implemented during the porting phase so newer projects can start with the new development environment.

The company has to setup their build pipeline between the version control system and the artifactory.

And the company has to setup the IDE and make sure that the IDE, version control system and the artifactory can all communicate together.

CONCLUSION

There are some steps that need to be completed before the switch can be made to a Cloud-native solution it starts by porting the code so that the code can still use their dependencies. The porting is done after the new artifact storage has been set up. From there on all the other systems must be set up, these are the following Version management system, a build/test system and an IDE.

As mentioned in the previous literature questions (chapter 6.3.4), Thales has the knowledge to fulfill a full migration.

7 RESEARCH CONCLUSION

All the previous research must come together to create the design principles. The design principles help define the choices made in a design, while not all the design choices have to be used in the design. The goal is that a design can be explained using the design principles. The design principles have a standard layout:

If (the problem) then (the solution) because (the reasoning).

DP 1 – If Thales want a new development environment that makes use of the newest technologies, then Thales should use a Cloud-native development environment, because the Cloud-native trend is relatively new and supported by world leading companies help design the newest techniques in an open-source fashion.

DP 2 – If Thales does not want a vendor lock on their systems, then a Cloud-native solution can help Thales because the CNCF is designed with the idea of multiple applications and micro services working together and have possibilities to integrate which each other.

DP 3 – If Thales want to save money eventually, then Thales should go for a Cloud-native solution, because Cloud-native solutions are built with the idea of scalability and can auto scale with the demand. This together with the fact that Cloud-native solutions are designed to be more efficient with the resources.

DP 4 – If Thales want to have a simple way to manage their systems, then a Cloud-native solution can help Thales, because Cloud-native application try their best with an ease of configuration mindset. Most Cloud-native solution have a way to connect another application to theirs with just an API.

DP 5 – If Thales want to have multiple ways to implement a new system, then moving to a Cloud-native solution can help Thales because there are at minimum 4 ways to choose from in moving their systems to a Cloud-native environment

DP 6 – If Thales want to have an on the shelf solutions, then Thales can consider a Cloud-native solution because in the Cloud-native landscape a lot of different applications exist which Thales can use to create their own solution.

DP 7 – If Thales is looking for an All-in-one solution, then Cloud-native can be a great option because there is great application in the Cloud-native landscape like GitHub and GitLab.

DP 8 – If Thales needs to have their environment in run locally then Cloud-native can help Thales because Cloud-native does not mean that it is only accessible in the Cloud, most Cloud-native applications can be run locally on a on premise server.

8 DESIGN

8.1 PREFACE

The design is the next step of the design research method. The design chapter is used to show what the new changes will be and show the differences between the current and new situation. Further this chapter will consist of a functional design and a technical design, furthermore the test environment will be described.

All the previous research will be used to create a design, the previous research will consist of the requirements (chapter “4.4 requirements” from the needs analysis), the conditions (chapter “5.4 conclusion” from the context analysis) and the design principles (chapter “7 research conclusion” from the research conclusion). The compliance of the previous research can be found in the evaluation chapter. The design chapter consists of the following chapters:

- Used methods – In this chapter the used methods will be described.
- Cloud-native solution comparison – In this chapter different solutions will be compared, and the applications that help build the environment.
- Functional design – In this chapter the current and new situations will be described using UML drawings.
- Technical design – The prerequisites for a Cloud-native development environment, architectural drawings for the new situation.

During the design phase some of the sub questions will receive valuable input. These questions are the following:

- What is needed in development environment for TACTICOS applications?
- What are the advantages and disadvantages of going Cloud-native?
- What are the requirements for a web-based development environment to be run on a cloud-native platform?
- What web-development tools are available on the market?
- How would the design and implementation of a web-based development environment be run on a cloud-native landscape?

These questions will help create and define a possible design. The results of the design phase will be used in the testing phase and the advisory.

8.2 USED METHODS

8.2.1 ARCHITECTURAL MODELS

In the design phase the goal is to make a design of the situation in the future. In the “context analysis” the same architectural model has been made, the difference between that one and the one made in the design phase is that the one in the design phase is made for a future solution and is based on the one from the “context analysis”.

8.2.2 UNIFIED MODELING LANGUAGE (UML)

The UML method will be used to visualize the functional level of the design. The current and future situation will be compared, whereas the current situation will be based on the context analysis and the future situation will be based on the research done so far.

8.2.3 PROTOTYPING

Another method used is the prototyping method. The prototyping method is used as the prototyping design approach mentioned in the Saxion “Design Research” methodology. During the prototyping method gives flexibility and allows for multiple solutions to be tested. Prototyping may include elements of systematic or deliberative approach, but it does not have to. (Haveman, Hageraats, & van der Linden, 2016)

8.3 CLOUD-NATIVE DEVELOPMENT ENVIRONMENT COMPARISON

For the Design phase 2 designs will be created, the designs will be prototyped and tested. For the all-in-one solution the top 3 options will be compared, while for the lift and shift method will be done with Kubernetes, the reasoning behind is that Kubernetes is the only solution out of the top 3 which either free to use or can be done with minimal resources for a prototype situation. However, container orchestrators use the same base functionality so if a situation works in Kubernetes, it has a high likelihood working in other container orchestrators.

The 2 designs will be based on the previously mentioned options a company has. The designs are based on what Thales has mentioned, the solutions architect named that Thales France is working with GitLab and that Thales itself is interested in working with their current systems but on a Cloud-native platform. So, for that reason the following 2 designs will be created:

- All in one solution
- Lift and shift

In the case that Thales is interested in an all-in-one solution with the support of other Cloud-native application they can add them. For now, the focus is on the core features of a solution. Thales also has mentioned that they are not interested in moving their current applications set to a new application set, so the “Individual Cloud-native applications” method is not going to pass

8.3.1 COMPARING ALL IN ONE SOLUTIONS

To start with the All-in-one solution. The goal of an all-in-one solution is to (almost) fulfill the full DevOps (Developer Operations) environment with 1 tool. A comparison will be made between the 3 biggest DevOps applications and the current environment, the 3 solutions that are chosen are GitHub, GitLab and Azure DevOps (Boards). According to Gartner these are the best alternatives for the current Atlassian based environment.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

	GitLab	GitHub	Azure DevOps	Current environment
IDE	<p>✓</p> <p>Basic features exist.</p> <p>Plugins available for all JetBrains IDEs and more.</p> <p>Extra paid service GitPod for web-based access to VSC and desktop access to All JetBrains IDEs and VSC</p>	<p>✓</p> <p>Basic features exist</p> <p>Paid service called GitHub Code spaces with advanced features</p>	<p>✓</p> <p>There is no web development tool, but Microsoft owns Visual Studio Code (VSC) as a separate application/License</p> <p>And offers great integration with VSC.</p>	<p>✓</p> <p>In the current situation Thales uses VSC in a Citrix environment.</p> <p>But different Cloud IDEs can be integrated in Bitbucket.</p>
Version control system	<p>✓</p> <p>Full GIT support</p>	<p>✓</p> <p>Full GIT support</p>	<p>✓</p> <p>Git support and Team Foundation Version Control support (makes use of workspaces)</p>	<p>✓</p> <p>Full GIT support in Bitbucket</p>
Code Building	<p>✓</p> <p>GitLab has built in features to build code and create artifacts.</p>	<p>✓</p> <p>GitHub has built in features to build code and create artifacts.</p>	<p>✓</p> <p>It has a feature called Azure pipelines.</p>	<p>✓</p> <p>Thales uses Jenkins for their testing/building.</p> <p>Bitbucket does offer a building solution</p>
Artifactory	<p>✓</p> <p>GitLab has an artifactory called GitLab Artifacts where artifacts can be stored.</p>	<p>✓</p> <p>GitHub has an artifactory called GitHub Artifacts which is like GitLab Artifactory</p>	<p>✓</p> <p>Azure has a feature called Azure Artifacts for storing artifacts.</p>	<p>✓</p> <p>Thales uses Nexus for their artifact storage. Here are all the artifacts stored after building the code.</p>
Price's	<p>GitLab server: price on inquiry</p> <p>Web version (GitLab</p>	<p>A similar package (GitHub) to GitLab Ultimate costs around \$21 per user/month.</p>	<p>Without any IDE the Azure DevOps basic + Test license costs:</p>	<p>Bitbucket+ Jira+ confluence should cost around 33,40 USD per person per month. Price is</p>

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

	Ultimate): \$99 per user/month. GitPod add-on: price on inquiry Or online version between \$8-35 a month.	GitHub Code spaces Or GitPod (see pricing at GitLab's pricing) First year between \$1299-\$2199 Second year 50% off if extended on time.	€52.01 per user/month An additional VSC license costs around €250 per user/month	calculated based on 250 users. Jenkins is free to use. Nexus needs a quote for 100+ users but should cost around \$100 per user per month. In total this plan will cost around \$133.40 per user per year.
Can be locally hosted	✓ GitLab Self-Managed	✗ Cannot be hosted in an on-premise environment.	✗ Cannot be hosted in an on-premise environment.	✓ Server solution is not supported anymore, Data center is still supported.

Table 13 Comparison between solutions

Advantages:

- GitLab
 - Can be locally hosted.
 - Lots of integration options from different applications.
 - Open source
 - Big community support.
- GitHub
 - Most advanced features.
 - Open source
 - Lots of integration options from different applications.
 - Big community support.
 - Base version is the cheapest per person.
- Azure DevOps
 - Great integration with Microsoft products.
 - Advanced features in the Microsoft ecosystem.
 - Microsoft support

Disadvantages:

- GitLab
 - By itself is it the most expensive solution.

- GitHub
 - The main components cannot be hosted locally. Just the runners to build/test the code.
- Azure DevOps
 - The main components cannot be hosted locally. Just the runners to build/test the code.
 - Not the best integration with other applications.

The 4 solutions in the comparison are considered the best alternative DevOps solutions compared to the current environment. If these solutions are compared with the design principles and requirements then the only enough solutions are GitLab, this main reason for this choice is the on-premise solution that GitLab offers and the other solutions do not. GitLab is also the only solution that fulfills all the design principles.

Further research will tell if GitLab is a better solution then the current DevOps solution

More details can be found in appendix "12.3,12.4 and 12.5"

8.3.2 COMPARING THE LIFT AND SHIFT CONTAINER ORCHESTRATORS

While there are multiple container solutions like VMWare Tanzu and Kubernetes. It is worth it to note that container orchestrators work nearly the same and a container application function on all of them. Kubernetes is considered the best open-source container orchestrator while the other 2 named solutions are paid solutions. Due to the standards in container solutions and the way containers are made can it be assumed that a solution in Kubernetes will also work with another container orchestrator.

8.4 FUNCTIONAL DESIGN

In the functional design the changes will be compared from the current system and the two designed solutions. The functional changes consist of two forms, the current and the future form. Both are based on the interviews and the context analysis.

MAINTAINER

In **the current situation** the maintainer does perform multiple tasks. The maintainer has access to the production environment, in there the following processes take place: update, maintain, monitor and troubleshoot. Updating the system consists of update the applications and the server. Maintenance is when configuration changes must be made, the server needs monitoring in case the environment starts to feel slow or storage runs out, also good monitoring software can help spot problems before they occur. The final process is trouble shooting, in case anything breaks the maintainer must fix it.

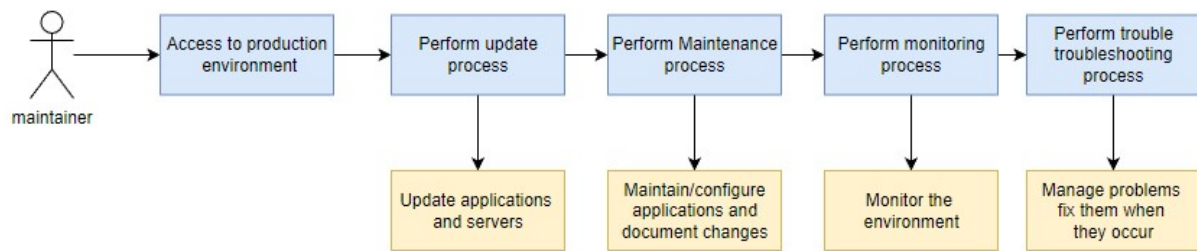


Figure 15 Current situation maintainer

The future situation will stay relatively the same, the time to perform some processes will slow down, this will be because Kubernetes has rolling updates and tries to restart shut down containers. These extra features will come to both solutions because Kubernetes is performing them.

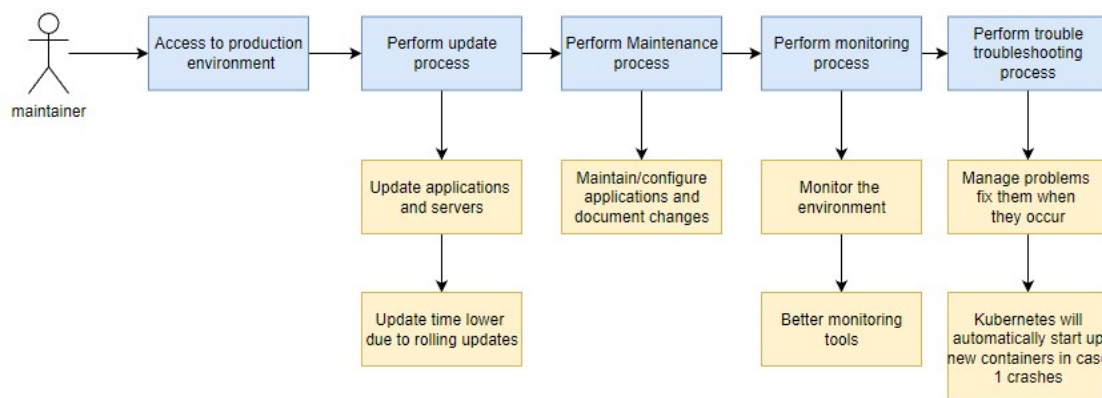


Figure 16 Future situation maintainer

DEVELOPERS AND TESTER

In **the current situation** the developers write the code and test their code. The first process is writing and updating code, the code gets build and tested in the second process. Any application will get released and stored in an artifact storage. The artifacts that are stored can be used in the future.

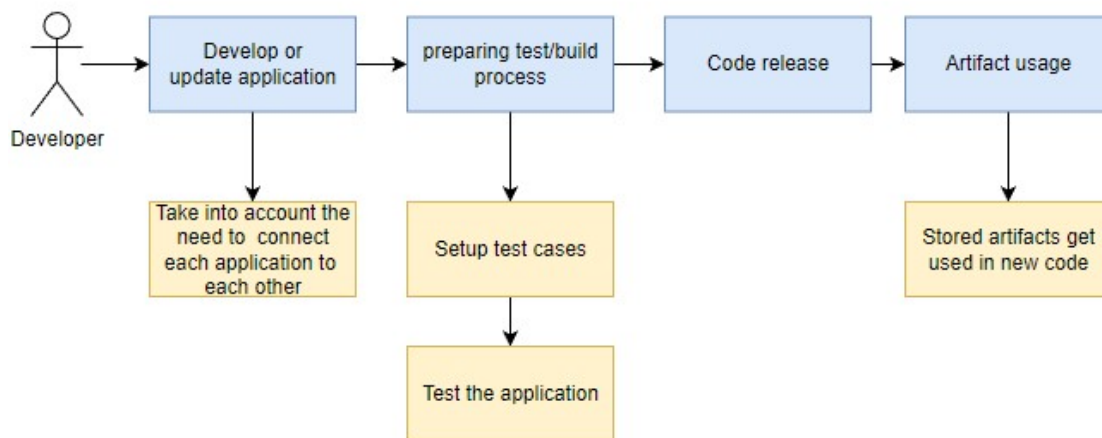


Figure 17 Current situation Developer

In the future situation the situation will only change for the GitLab environment. The lift and shift are the same situation on a different platform. The GitLab solution is better automated, saving time and time in different processes in the solution, GitLab's tests are more advanced. The core processes are the same.

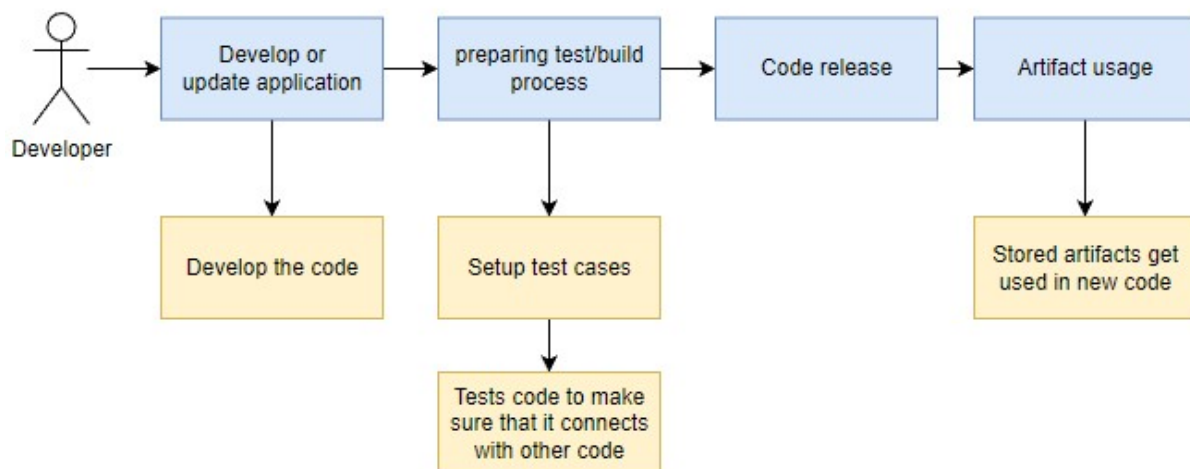


Figure 18 Future situation with GitLab developer

8.5 TECHNICAL DESIGN

The technical design outlines how the technical implementation of a Cloud-native development environment will work in the current environment and serve the use cases from the functional design. The prerequisites for a Cloud-native development environment will be described. A view enterprise architecture for a possible future solution will be shown.

8.5.1 PREREQUISITES FOR CLOUD-NATIVE DEVELOPMENT SOLUTION

There are no real prerequisites to implement a Cloud-native development solution. Implementing a solution can be done on every Linux distro. For this assignment tests will be done on the latest Ubuntu version to demonstrate that it is possible to run it Cloud-Native, tests will be done on the web-version.

Installations for both solutions can be realized using the created Guides/manuals (Chapter 12.8 Guides and manuals)

8.5.2 DESIGN CHOICES FOR THE FUTURE SOLUTIONS

The designs have been drawn with the design principles mentioned in the design principles (Chapter 7 “Research conclusion”), and the requirements (Chapter 4.4 “Requirements”). The design is firstly based on a Cloud-native fundament, offers less maintenance, both designs use on the shelf software and applications. So, the design can be used for other companies as well and for the GitLab design the choice was to make it an All-in-one solution. Both drawings have an optional part worked out for added features.

All the “Must” requirements are used in the choice for these solutions as they are the most important. Both the GitLab and the Lift and shift solution satisfy the requirements set. Some “should” requirements are added in the design by default due to the solutions offering the function needed to meet the requirement.

8.5.3 ENTERPRISE ARCHITECTURE CHANGES

The drawings have been split into 2 different environments, the maintainer side and the developer side. There are drawings for both solutions where one method is designed with the lift and shift method in mind the other with an all-in-one solution in mind.

LIFT AND SHIFT SOLUTION (KUBERNETES)

Both drawings can be found in appendix “12.2.3 New lift and shift solution”

This design is nearly the same as the current situation. The biggest change is in the technology layer, where everything is run in containers and were Visual Studio Remote development. This plugin in VSC is gives the opportunity to offload the workload in a

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

Citrix environment and moves that to a container this is to save locked in resources from the Citrix servers. In this case the server resources can be used on different servers.

For the maintainers most of it is the same, there are 2 applications added to the maintainers pool, these are remote VSC and Kubernetes.

ALL IN ONE SOLUTION (GITLAB)

Both drawings can be found in appendix “12.2.2 New GitLab solution”

This solution brings a lot of changes, within the drawing there is an optional section which is an option which for now focusses on a better coding solution. With the GitLab solution each member of a developing team can do everything they need in GitLab. The technology layer is also trimmed down to just a server with GitLab and an optional Citrix server to create a better coding environment (can be improved with Linux laptops or other solutions).

For the maintainers there is just 1 application that needs to be maintained, this makes it easier to manage/maintain, the application itself can take a while to setup as the current functions need to work in a new application.

8.5.4 THE TEST ENVIRONMENT

The test environment will be used to test both solutions within the possibilities. The test environment consists of 1 physical machine for both solutions. Both solutions run on Kubernetes and both run in a different namespace within Kubernetes.

The GitLab namespace consists of all the containers needed for GitLab. The Atlassian namespace consists of the current situation in a Kubernetes environment. A small disclaimer is that the current situation is not fully testable due to licenses, however the applications are the same as Thales uses in the current situation with the only change being set-up on a Cloud-native platform. Functionality of the application will stay the same, but Kubernetes offers autoscaling functionality. Which should help Thales in saving resources.

GitLab offers a free trial to use with limited functionality, so while not all the functions will work, the functions which cannot be tested work on paper.

The demo environment will use Helm charts to help install the needed container. Helm is a package manager for Kubernetes, this choice is made due to the time it will take to build an own container for Kubernetes. For Thales it might be an option to customize their own containers. The tests performed are functional tests to check if the design choices can be brought to reality and perform the functions that Thales wants.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

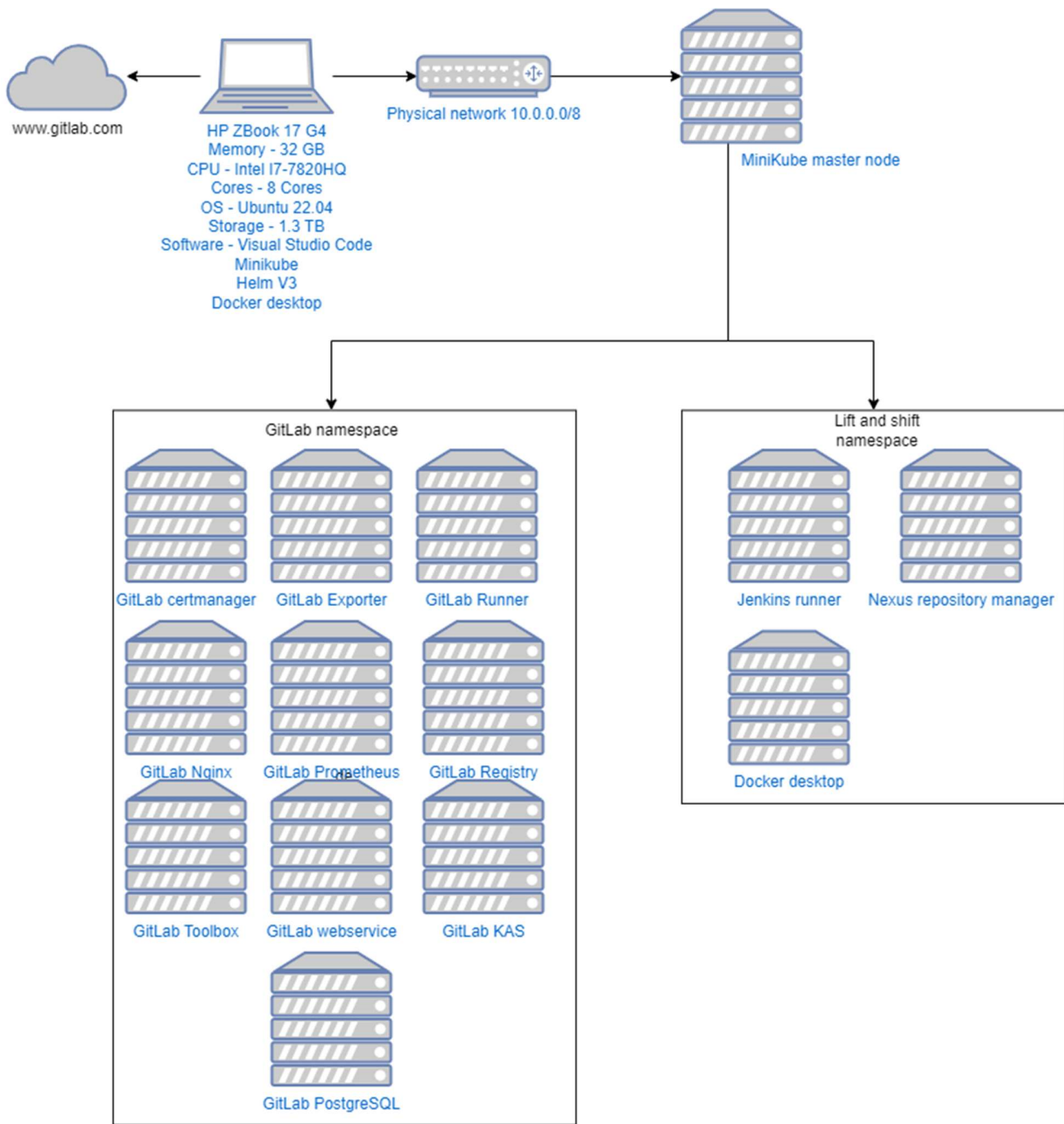


Figure 19 New development environment test environment

9 TESTING

9.1 PREFACE

Testing is the next step after the design phase is “testing”. The purpose of this chapter is to outline the topics and areas that will be tested. Developing a final judgment and

recommendation while taking the prior findings into consideration is a crucial step. During this phase the two chosen solutions (GitLab and Kubernetes) will be prototyped. After they will be tested following a procedure of tests based on the requirements set earlier. The following chapters are part of the testing chapter:

- Used methods – Describing the used methods during the testing phase.
- Tools and applications – Describing the applications and tools used for testing.
- Test environment and procedure – Describe the test environment and the testing technique.
- Test conclusion - contains the test results and a conclusion drawn from the results.

During the testing phase, a few sub questions will return answers and provide feedback. The following questions are:

- How would the design and implementation of a web-based development environment be run on a cloud-native landscape?
- What is a web-based development environment?
- What is needed in development environment for TACTICOS applications?

These questions are important to the main question and are essential to the final advisory. This advice is primarily based upon the test results. The tests also comply to specifications, rules, and design principles. The evaluation chapter contains the results of these.

9.2 USED METHODS

9.2.1 PROTOTYPING

The Prototyping Design Approach, as mentioned in the Saxion "Design Research" methodology, is chosen for the design. This approach allows for the creation and testing of multiple solutions, which is an essential element of this assignment. The researcher has used one of the three approaches mentioned in the "Prototyping approach"

9.2.2 FUNCTIONAL TEST

The functional test is used to test the requirements, the goal is to test the functionality of the prototypes and see if the functionality matches the requirements the customer set during the assignment. These requirements focus is about functionality. This method is tested using the applications and see if they can run on a Cloud-native platform and perform the functions that the customer requires.

The functional test is testing the mainline functions and accessibility of an application. Having the applications perform the main tests

9.2.3 DESK RESEARCH

Desk research will be used to gather information about installation guides, answers to certain questions that during the set-up and tests and read about others user experience. More information in chapter “Desk research”

9.3 TOOLS AND APPLICATIONS

The perform the tests in the test environment some applications and tools have been used.

The first tool that has been used is Helm. Helm is a package manager for Kubernetes helping this tool helps managing the applications and making sure Kubernetes does what it must do. Helm also makes it possible to configure applications before they launch. Helm charts, the main component of Helm gives the researches the option to read installation/configuration settings.

To test GitLab's functionality the web version of GitLab is used*. The web version offers the same functionality as the self-managed version and will be used to test the in-application functionality.

The first application being used is Visual Studio Code (VSC), this application is being used together with the remote development plug-in available in the plug-in marketplace. Gives this application the remote development possibility.

*This is due to free license not being accepted in the self-managed version.

9.4 TEST ENVIRONMENT AND PROCEDURE

A test environment is designed during the design phase and will be built and tested. The "Test Environment" chapter has details on this setup. The tests regarding the systems will be done on the test laptop, the tests regarding the functionality of an application are done using the web version.

The prototype environment meets a few set requirements to make sure that the results are consistent.

- Each solution is setup using Helm as a package configuration.
- Each solution is setup with at least 1 load balancer service included to make a connection possible.
- The web version of GitLab makes use of a free trail license.
- All environments are setup in a different namespace.

Some key points worth mentioning

- Each test setup is used using the steps from the “Guides & manuals” step.
- For the functionality test of the lift and shift environment it can be assumed that all functions work as they do in the current environment, so the important

part is to check if the environment can be run on a Cloud-native environment.

The following test setups have been performed.

- Deploying an application on a Cloud-native platform:
 - GitLab environment
 - Using Helm, the deployment of all the applications needed to run GitLab will be deployed on Kubernetes showing that the environment can be run on a Cloud-native platform.
 - Lift and Shift environment
 - Using Helm, the deployment of all the applications needed to run Bitbucket will be deployed on Kubernetes showing that the environment can be run on a Cloud-native platform.
 - Using Helm, the deployment of all the applications needed to run Jenkins will be deployed on Kubernetes showing that the environment can be run on a Cloud-native platform.
 - Using Helm, the deployment of all the applications needed to run Nexus will be deployed on Kubernetes showing that the environment can be run on a Cloud-native platform.
 - Coupling Visual Studio Code to Docker to start up a container in Kubernetes to run/debug code.
- Testing the IDE
 - GitLab environment
 - Opening the web version and write test code, debug it and check for plugins usage.
 - Lift and Shift environment
 - Opening the application write test code, debug it, check for plugins usage and if the code gets run in a container.
- Testing the version control system
 - GitLab environment
 - The previous written code will be committed, then changed and again committed to see the changes made.
 - Lift and Shift environment
 - Bitbucket will be deployed using helm on a Kubernetes platform and tested to see if it works.
- Testing the pipeline/build tool.
 - GitLab environment
 - Check if the runners are functioning in the pipeline.
 - Lift and Shift environment
 - Check if Jenkins can be run in a Kubernetes environment.
- Testing the Artifact storage.

- GitLab environment
 - The pipeline jobs will push the artifact automatically to the artifact storage.
- Lift and Shift environment
 - Testing if Nexus can be run on a Kubernetes environment.

These tests will make sure that the functionality of the application can be performed and help find an answer for Thales to see if a Cloud-native solution is an option that Thales can rely on.

9.5 TEST CONCLUSION.

Unfortunately, not all tests could be performed. To start with GitLab had a big web ide update on December 19 2022, which disabled the run/debug solution and the marketplace. These features do exist and will be brought back. Another problem was that the trial license blocked the use of the pipelines/artifact storage. Again, these functions exist.

In the lift and shift environment Bitbucket was the biggest problem, it required a license key configured into the code before actually being able to be deployed. So, on it should work as Atlassian offers their own Helm chart and guide on how to deploy Bitbucket.

If we look at the IDEs the tests were a success, GitLab with the new web ide update offers a coding experience that looks and feels like Visual Studio Code. It will offer a market place soon and all the core functionality Thales requires in their code. GitLab also explained that they keep working on improving the web IDE.

The remote code solution that Visual Studio Code offers works as a great solution to offload the resources from a Citrix environment of laptop to a server backend that spins up a container to run the code.

The version control system of GitLab works as intended, showing all the versions, who committed the code, what has been updated and more. As mentioned before Bitbucket could not be run due to the license and not wanting to spin up a container without a license, but Atlassian does offer an official guide and all the official Helm charts needed to setup an environment in Kubernetes.

The next thing that has been tested is the Pipeline/build function. As mentioned in the first paragraph this was not an option to be tested in GitLab, the function in GitLab is called GitLab runner. It offers a scalable system in Kubernetes as it will deploy a container to build/test the code and closes the container after the building/testing is done. In the lift and shift environment it worked as intended, Jenkins is an open-source

solution which offers a great integration with version control systems and artifact storages. It also offers the same scalability as GitLab.

The final test is about the artifact storage. Again, as mentioned the artifact storage from GitLab is not available due to the licensing. GitLab mentions its functionality and explains how it functions on their website and forums. Nexus also has full support from their vendor Sonatype, making sure there are official Helm charts and a guide on how to setup their environment, Nexus does function well on a Cloud-native environment.

Not all requirements were able to be tested to the full extent, some functions were limited to a full application. Plugins that were not testable in the GitLab environment due to it being temporarily disabled.

All the test results can be found in the appendix "Testing results"

10 EVALUATION

The goal of this chapter is to evaluate the various products and phases within the thesis. The main and sub questions are answered, and check if the solutions comply with the requirements and conditions. The evaluation is divided into two parts: research evaluation and design evaluation.

10.1 USED METHODS

The following methodologies are used in the evaluation:

- Verification with the stakeholders
 - The results of the tests and the evaluation will be discussed with the primary stakeholder “Willy” and decided if everything is as the company wishes.
- Prototype testing
 - The test environment is built and tested to gather results. These results are in the appendix “testing results”.

10.2 RESEARCH EVALUATION

The research evaluation consists of 3 products, these will individually be evaluated:

- Main and sub questions
- Requirements
- Conditions

10.2.1 MAIN AND SUB QUESTIONS

The full evaluation can be found in appendix “12.7.1 main and sub questions evaluation”. In the appendix is a reference to each question and a summarized answer can be found. All the (sub)questions have been answered during the research and concluded in the evaluation.

The main question: “**How could a web-based development environment for TACTICOS applications be implemented on a Cloud-native platform?**” gained the following answer:

A web-based development environment can be implemented using multiple strategies, there are two designs chosen on which one is an all-in-one solution and the other a lift and shift solution. Both were able to keep the same functionality a normal development environment has. This combined with better maintenance tools extra configuration options in the Cloud-native platform, makes a web-based development environment worth it.

10.2.2 REQUIREMENTS EVALUATION

All the requirement has been evaluated in appendix “12.7.2 requirements evaluation”, each requirement has been checked using the tests and the research. Each requirement has a description on why it passed/(partly)failed. It is important to mention that GitLab recently received an update, due to some features being temporarily locked.

10.2.3 CONDITIONS EVALUATION

The entire evaluation can be found in appendix “12.7.3 conditions evaluation”, in there is a description for every condition. Only 1 condition did not pass due to it not being able to test it.

10.3 DESIGN EVALUATION

In the design evaluation the design and tests will be evaluated in the following products:

- Design principles
- Testing

10.3.1 DESIGN PRINCIPLES EVALUATION

All the design principles have been evaluated and can be found in appendix “12.7.4 Design principles evaluation”, all the design principles have been evaluated and have been described on why and where they are used.

10.3.2 TESTING EVALUATION

Setting up the test environment brought a few small problems; the problems occur due to not having paid licenses, however the answer to the possibilities has been found due to the big community support and the support from the vendor an answer was given on all tests and the requirements that had to be met.

The other tests were successful and went without any problems. The applications did what they were supposed to do, the current situation can be run on a Cloud-native platform, while GitLab can be a strong contender of even a better solution.

The testing results are comparable with the literature the full testing evaluation is in appendix “13,4.5 Testing evaluation”.

11 ADVISORY

If Thales were to make the step into designing a Cloud-native development environment, then Thales still has a lot of options to choose from. A Cloud-native development environment can be an improvement over the current environment, this is due to higher efficiency. The higher efficiency equals to lower resource usage for the same or better performance. However, some application are not completely finished and might not be optimized.

A Cloud-native solution may also bring more customization opportunities, because of simple application integration and new solutions keep entering the market bringing new advanced features. Looking at the solutions there are still a lot of possibilities, but a recommendation is to wait a few years so that the web IDEs can catch up with the other parts of a development environment.

The Cloud-native market is dynamic and new solutions and features come out regularly. During the graduation period JetBrains (IntelliJ) announced a mayor update to their remote development solution. While VSC (Visual Studio Code) has a great VSC server solution which makes sure the code is run on a remote server (still need to be tested), but in the current situation the IDEs are the most behind. While the IDEs mentioned are from 2 of the biggest companies that are active in the IDE world. Some smaller companies already have true Cloud-native IDEs, others have a web development environment. But there are a lot of different applications.

On the other hand, if you look away from the IDE and look at the different segments of a development environment then there are great solutions, like GitLab as an all-in-one solution. A company also could customize their environment to their specific needs using different Cloud-native applications which as mentioned have great integration with each other.

So, now if Thales were to go with this moment Cloud-native solution for their development environment, then GitLab will fit Thales their needs the best. However, the recommendation is to wait a few years to let the web-IDE part of both solutions catch up with the applications. Both solutions offer web-based development and can be implemented on a Cloud-native platform. Almost all the requirements have been met during the tests.

11.1 RECOMMENDATIONS

Not all the solutions have been researched so some recommendations can be given for any future research or on what to keep your eye on.

- First, keep an eye on GitLab, if it is considered not the best solution at this moment keep an eye on it, GitLab is continuing to improve and keeps pushing new functions with a great roadmap on their website showing what they are working on and what they still want to add.
- Another recommendation is to test the solutions on a greater scale inside the company with a copy of the current system. Test it with developers over a certain time.
- Another recommendation is to do research into a full open-source Cloud-native solution.
- Keep an eye on future Cloud-native technology.

12 APPENDIXES

12.1 INTERVIEWS

12.1.1 INTERVIEW WITH WILLY

1. How do you think a cloud-based development environment can help Thales

Thales must obey to some strict rules on how our applications are run and how our data is stored. One of these rules is that all data/applications must be hosted locally. So, while there is a chance that your solutions will not be used but Thales need backup solutions in case that the current vendors decide to drop the support for their self-managed solutions and only support their Cloud solution.

2. Can you think of any problems where a Cloud-native development environment can help?

First, a cloud-native solution should bring more scalability to the system.
Secondly, a solution can help Thales save resources.

3. Are there some requirements you can think of that will be relevant for this Project?

A design must be hosted locally.
Kijk nog even naar de requirements

4. What is Thales already doing with Cloud-native?

Thales has been working with Kubernetes for a while for specific systems, but not yet for internal/own use.

5. Does Thales have any plans in going for a Cloud-native solution?

Not that I currently know of, this assignment will be the first for Thales Hengelo. I know that our French colleagues use GitLab for their entire development environment so that could be a start.

6. What results do you hope to get out of the research?

I hope that you can come up with some strong arguments on why a Cloud-native solution can be sustainable for Thales.

7. Can you name some stakeholders who also might be interested in an interview?

Rick Jansen is a solutions architect at IS/IT (Thales internal IT), he also has a lot of knowledge of the current system.

Daniel van Gils is a Cloud-native architect/systems architect who has a lot of knowledge of Kubernetes and other solutions.

Rob ten Hove is a Software architect for our CMS system and can help you with the software engineering perspective.

Lex Emmens as a software engineer. I recommend to go for Rob ten Hove but if he is not available ask Lex for an interview.

12.1.2 INTERVIEW WITH RICK JANSEN

1. What does a solutions architect do?

A solutions architect is concerned with maintaining and improving current environments. They do this together with all teams from IS/IT (Thales own IT).

2. How are the current applications hosted in the system? (Nexus, Jenkins, Bitbucket and the IDE)?

Currently, the entire development environment is built on different servers that use virtualization, so all applications are independent of each other and not connected to the Internet. All applications are managed on the server itself apart from the IDE, which runs in the Citrix environment and is updated using Citrix tooling.

3. How do you think moving the development environment to a Web-based development environment could help Thales?

A big advantage is that you have all the Cloud-native benefits such as high availability and scalability. In addition, you have an advantage that an environment is also easier to manage.

4. Can you think of any potential drawbacks that might apply to this?

A big disadvantage that comes into play but especially at Thales is that there are many diverse environments, because there are so many developers and there are also many programming languages used, each environment is different. So, application A may work for group A but application B may not work for group A and group A probably does not want to switch either.

5. What do you think are requirements relevant to a web development environment in the cloud?

We must be able to host the environment locally.

The environment must have high availability.

6. Do you know if an entirely different solution has ever been looked at? Like Gitlab or GitHub?

Yes, Gitlab has been looked at, but right now Thales Netherlands is very much in the Atlassian world. Therefore, the problem is that switching to a completely different environment is difficult.

However, it is being looked at as a backup option for when Atlassian no longer supports the private data center option.

12.1.3 INTERVIEW WITH DANIEL VAN GILS

1. What does a Cloud-native architect/systems architect do?

I keep myself busy by designing the Cloud-native side of my project group, besides I am helping other project groups and Thales with making Cloud-making choices and bringing my knowledge to help decide if a choice must be made and how an environment should look like.

2. How are the current applications hosted in the system? (Nexus, Jenkins, Bitbucket and the IDE)?

Basically, all applications except the IDE run on physical servers as far as Daniel knows. Essentially, the IDE is run in a VDI environment which is already a virtual desktop. But a physical application is still started.

3. How do you think moving the development environment to a Web-based development environment could help Thales?

It can help Thales because you have a portable environment and all dependencies and packages start right when you open the code.

The environment can be used anywhere

With some web IDEs you have many plugins which makes the web ide very diverse.

The IDE can also run in a container.

4. Can you think of any potential drawbacks that might apply to this?

Well, it depends on what type of solution you come up with, in the case you go for a container solution you will get different drawbacks then if you go for a GitLab/GitHub solution. While I am not up to date on the current GitLab version from experience I can say that the IDE was missing some core functions like debugging tools.

5. So you have investigated GitLab?

Yes, Gitlab was tried several years ago but the web IDE was not good enough at that time, in addition some functionality was missing in the application at that time.

6. What do you think are requirements relevant to a web development environment in the cloud?

For the development environment, make sure you keep in mind the core functionality of the IDE. If these are missing then it is soon not good enough for a developer.

12.1.4 INTERVIEW WITH ROB TEN HOVE

1. What does a software architect do?

A software architect focuses on all the software projects at the same time and makes sure that what they deliver will work together with the other code.

2. How do you think moving the development environment to a Web-based development environment could help Thales?

New developers can get started quickly because the IDE does not need to be configured first and no license needs to be attached.

Peer programming, just go to 1 session together see each other's code and program together.

The most important thing is that everything can load very quickly. Now it is always a bit of searching for where the code is and everything that goes with it. This could ensure that you can open your IDE at once and everything is there.

3. Can you think of any potential drawbacks that might apply to this?

So far actually more advantages than disadvantages, these mainly have to do with the current plug-ins that are now widely used in addition to the IDE. There is also a fear that the environment will become slow if there are many users using it at the same time.

Also, some developers like to use a different IDEs some do not support a Cloud-native solution.

4. What do you think are requirements relevant to a web development environment in the cloud?

Possibly shielding parts of the development process for shared files with colleagues from a different country.

Speed, the speed is important, that it does not collapse when more people are using it.

Functionality is also important, especially for the IDE, our developers need to debug and use plug-ins to help them write code.

5. What is your opinion on different environments like GitLab and GitHub?

From a developer point of view is GitHub one of the best and most used tools, while I am not known with the full capabilities of GitHub and GitLab. I understand the basics but have never used the full capabilities.

12.2 ARCHITECTURE DRAWINGS

12.2.1 CURRENT SITUATION

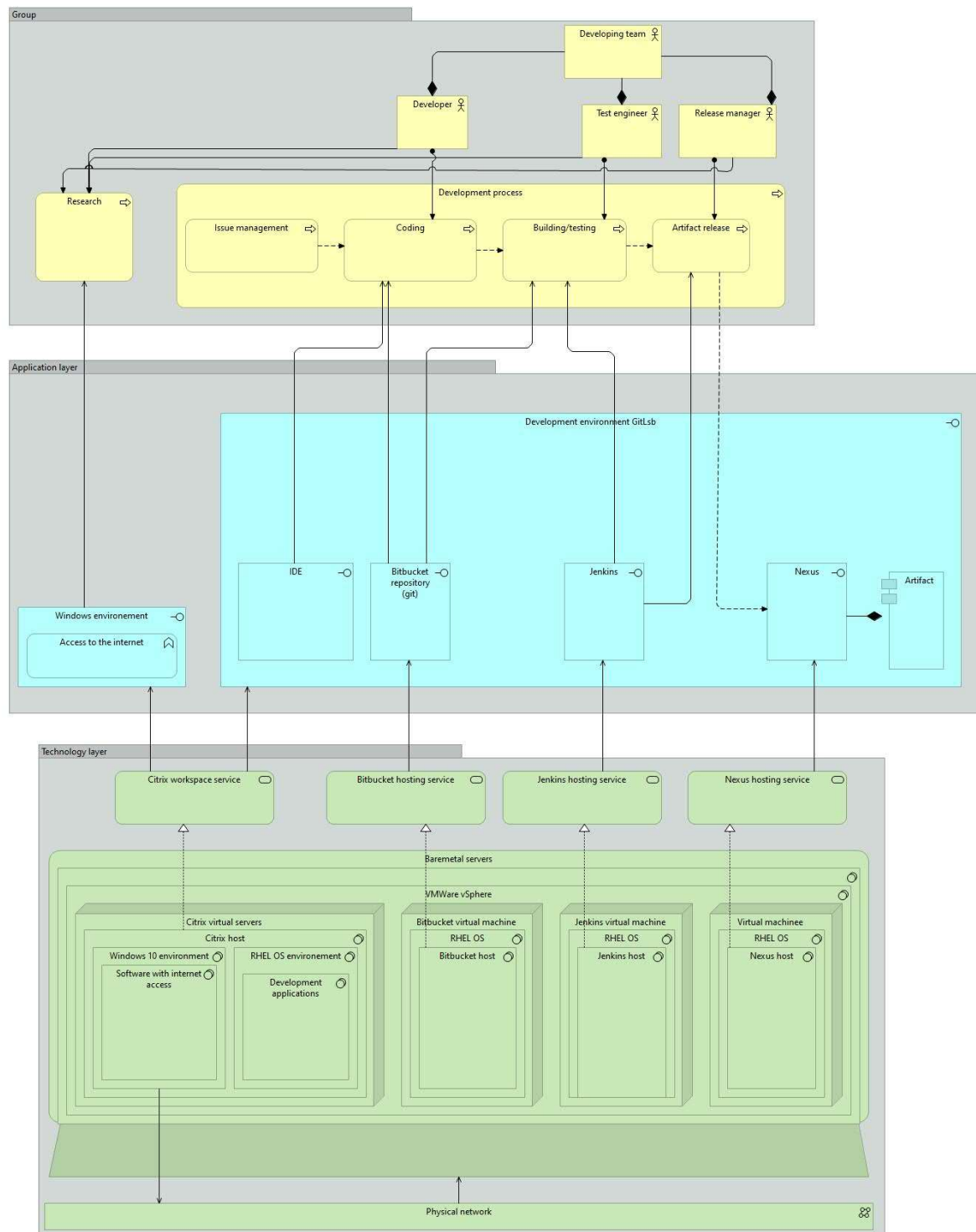


Figure 20 Architecture drawing current developer

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

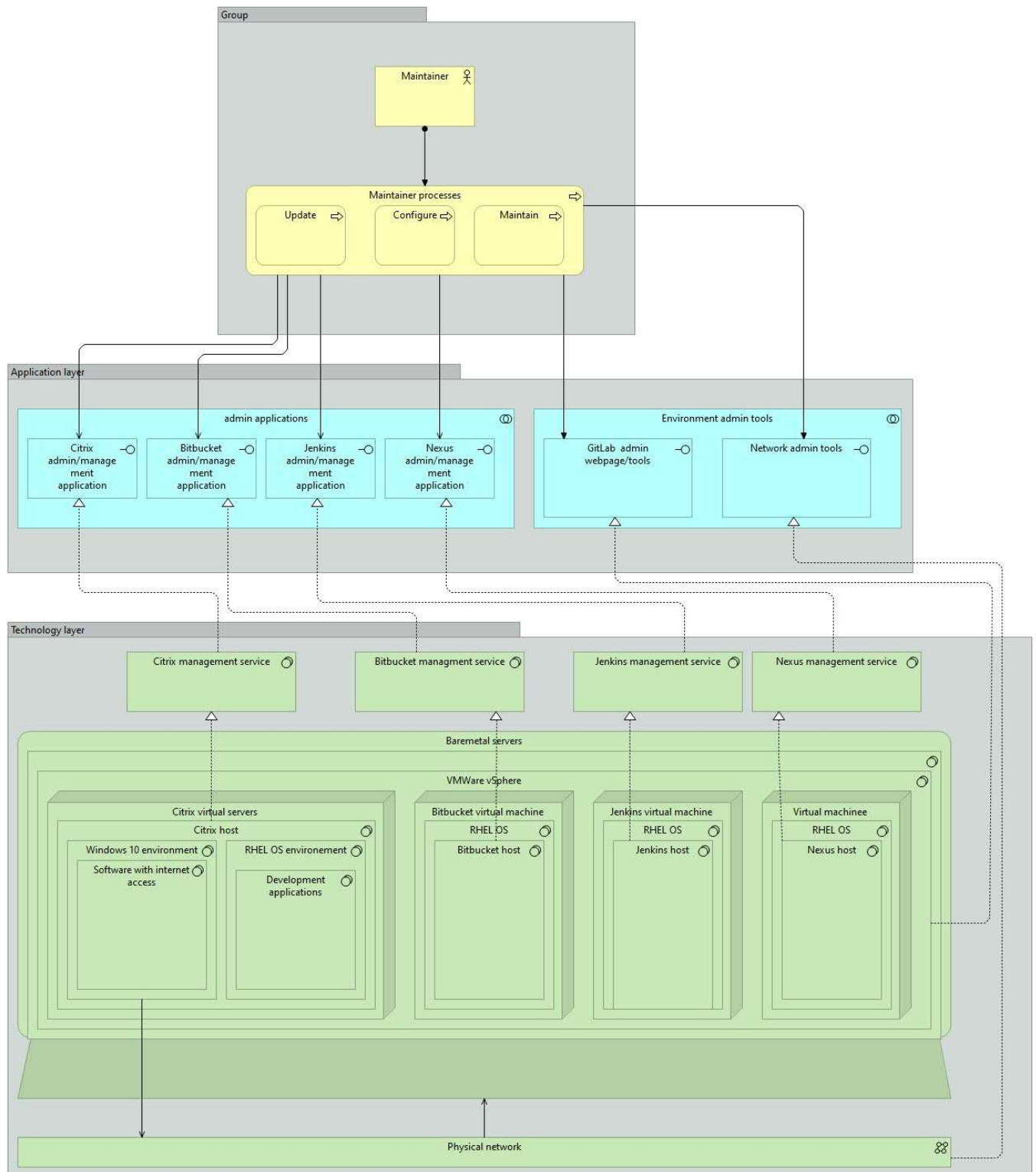


Figure 21 Architecture drawing current maintainer

12.2.2 NEW GITLAB SOLUTION

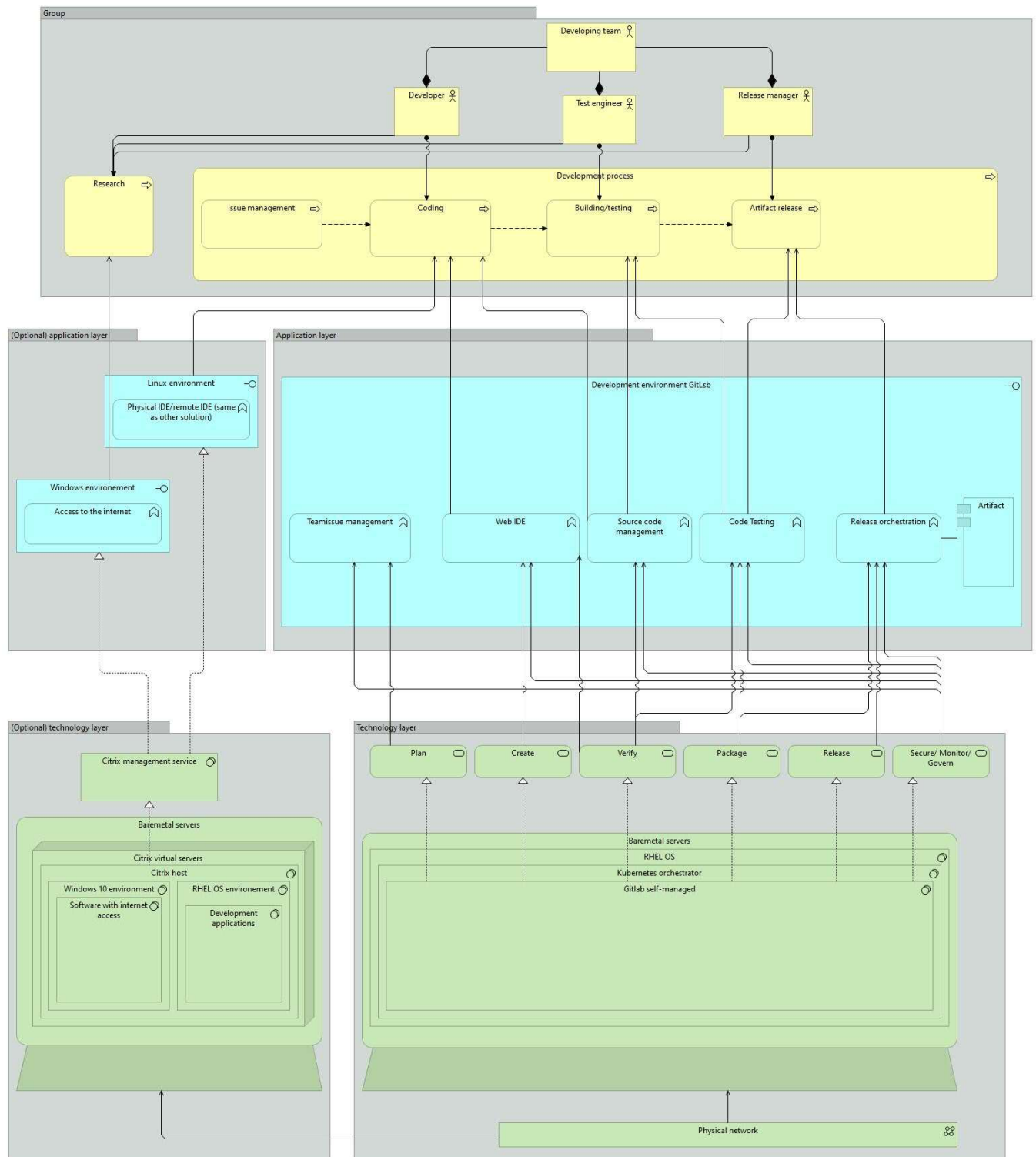


Figure 22 Future development environment GitLab

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

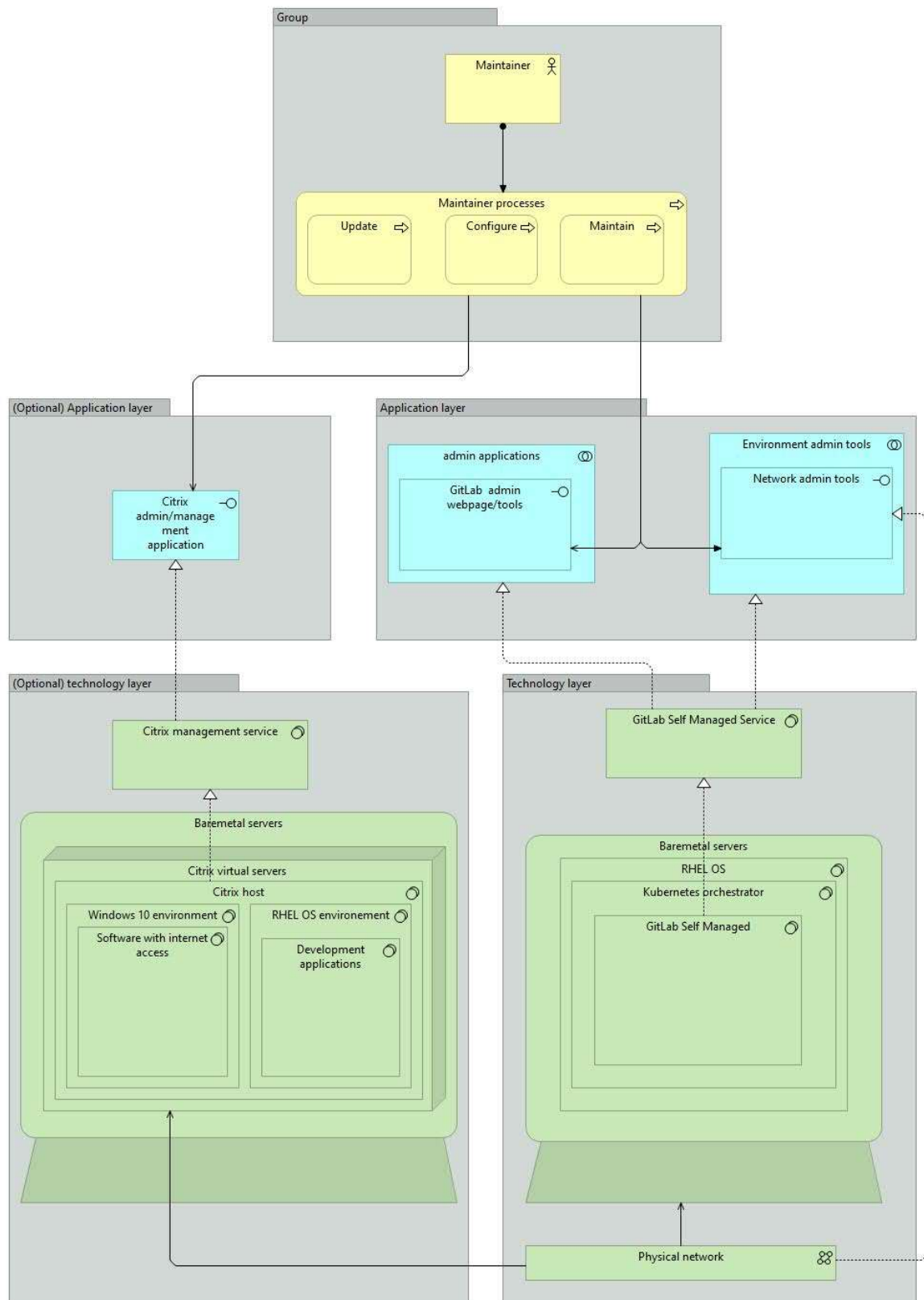


Figure 23 Future maintainer environment GitLab

12.2.3 NEW LIFT AND SHIFT SOLUTION

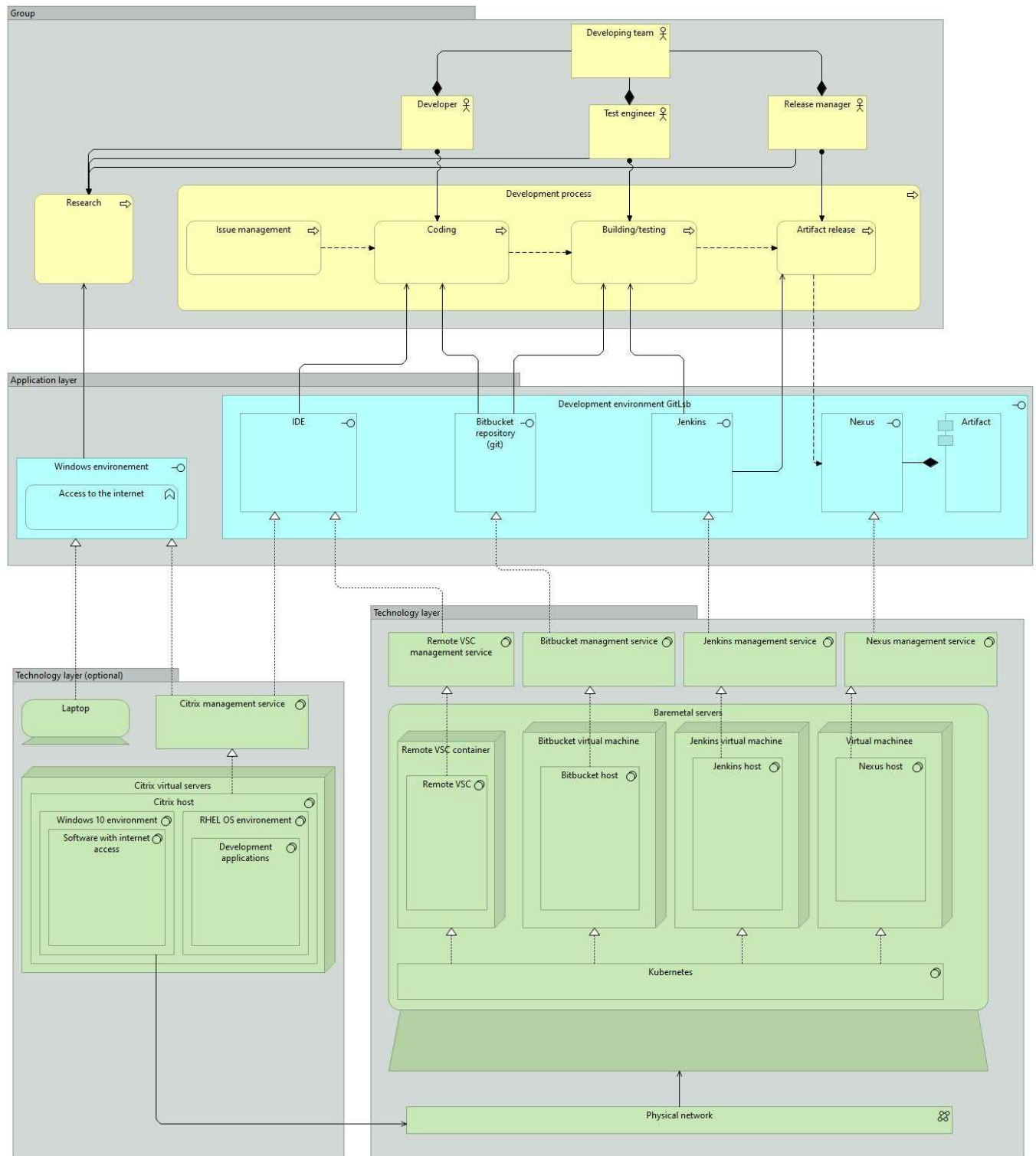


Figure 24 Future Development environment Kubernetes

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

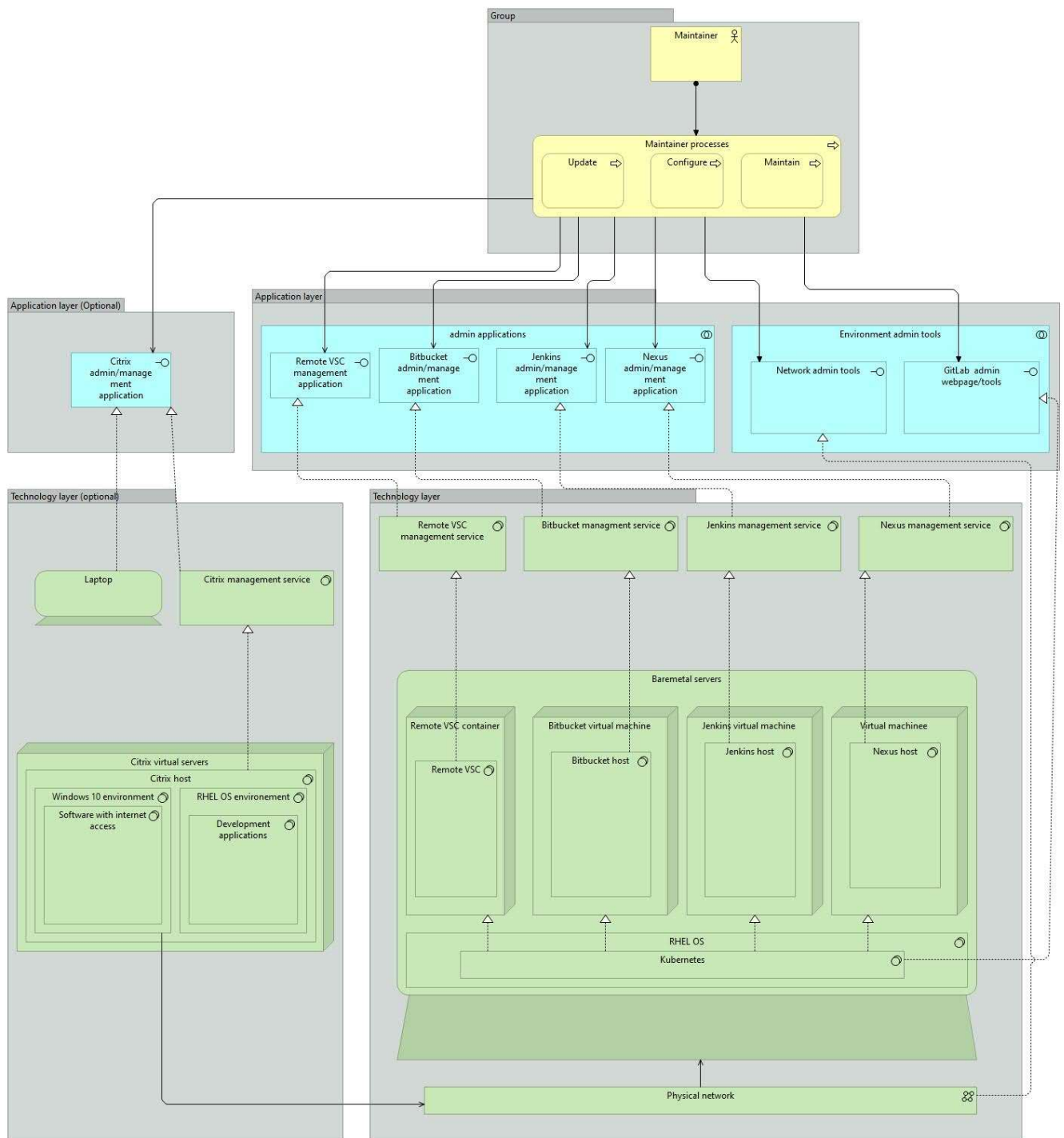


Figure 25 Future Maintainer environment Kubernetes

12.3 CLOSER LOOK AT THE GITLAB SOLUTION

GitLab is one of the best-known all-in-one web-based development environments, GitLab is an open-source project and backed by industry leading companies like Nvidia and T-Mobile.

GitLab offers a complete development environment, from writing code to deploying code it can all be done within GitLab. A company can easily change the GitLab environment to their liking, adding other applications to it and many ways to deploy GitLab itself. GitLab's main features are focused on the DevSecOps. The term "DevSecOps" is a combination of "development," "security," and "operations."



Figure 26 GitLab logo

The picture below shows the 9 main features and how below is how they support the features and in what stage of development they are.

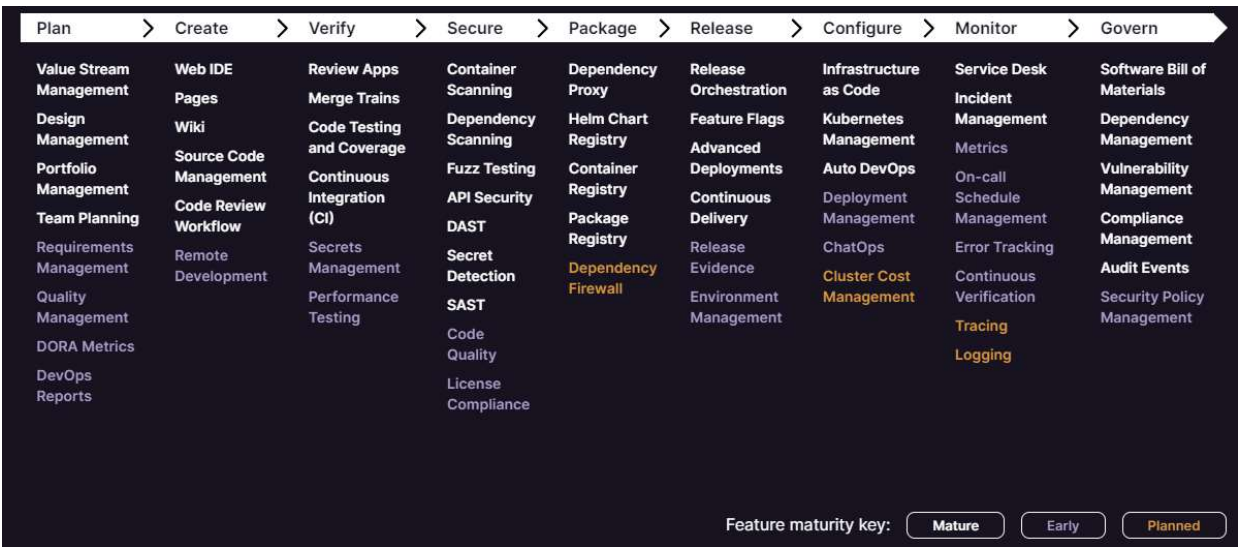


Figure 27 GitLab features (GitLab, 2023)

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

He features which are the most relevant to the project are “Create”, “Verify”, “Package” and “Release”

- **Create** is focused on writing the code and the code repository and merge requests.
- **Verify** and **package** are mainly focused on building and testing the code.
- **Release** is focused on storing the artifacts.

Together they offer a complete development environment. GitLab is part of the Cloud-native Computing Foundation which support the Cloud-native environment.

GitLab's wide variety of possibilities makes it a perfect application to use as a development environment.

GitLab offers roles to users and guest. This can be used to give a different facility access to the specific part of code they should have access to without them being able to see the other code.

GitLab offers issue management within the application, so developers can always see what must be done per project they have access to. Once the code is ready to be build it can be built, tested and reviewed in the GitLab pipeline.

Build – Code will be built so it can be tested.

Test – GitLab performs a few default tests like code quality and dependency's, it also offers a company to develop custom tests for their application. GitLab offers a static and a dynamic application security test, for errors in the code, errors like known code vulnerability errors and cross-site scripting vulnerabilities.

Review – In the review stage of the pipeline a test deployment will be deployed so the developer can test the application before releasing it. The application deployment will also get an extra dynamic application security test to find cross-site scripting vulnerabilities.

DEPLOYMENT OPTIONS FOR GITLAB

GitLab offers 3 solutions to be deployed. The first solution is the web solution. GitLab hosts the environment and everything that is part of it. This solution is great for smaller companies that do not have the infrastructure to host their own servers.

The second option is to deploy it straight on Linux, this is a simple installation on a server.

The third option is to deploy it on Kubernetes, this deployment method brings Cloud-native functionality to the application.

INTEGRATION POSIBILITIES WITH GITLAB

A Gitlab environment is adaptable to any sort of project. There is an extensive list off applications which can be integrated in any project. The reason a company chose to integrate different applications into a project is that some applications bring special features that can be useful. The list in the following picture gives an indication of which applications you can integrate:

Project integrations

Asana

Bamboo CI

Discord

Emails on push

GitHub

Google Chat

Harbor

Irker

Jenkins

Mattermost notifications

Mattermost slash commands

Microsoft Teams

Mock CI

Pipeline status emails

Pivotal Tracker

Prometheus

Pumble

ServiceNow

Shimo

Slack notifications

Slack slash commands

Slack application

Unify Circuit

Webex Teams

Webhooks

Some of these applications are well known like Jenkins, Slack and GitHub. But applications like Prometheus are used for better monitoring.

Another application mentioned inside of the project is GitPod, GitPod is not in this list but can be used to improve Web-IDE functionality, it will add extra costs to an environment.

Figure 28 Applications integration GitLab

12.4 CLOSER LOOK AT GITHUB

GitHub is considered market leader in the world of web CI/CD environments. GitHub offers almost the same functions as GitLab, both solutions try to copy each other but GitHub their IDE is more advanced. It already offers the functionality that GitLab is trying to implement, functions like code debugging and an application marketplace.

The version control system is based on GIT, so code must be committed to a branch. GIT is a well-known version control method. Which is used by all other possible solutions (GitLab, Azure DevOps).

Another important part of the developer operations is building and testing the code. GitHub offers a feature called GitHub Actions, GitHub actions offers a few features called "Matrix builds", this feature gives the developer the option to start multiple tests at the same time across different operating systems. This function will lower the build and test time.

GitHub has been taken over by Microsoft in 2018



Figure 29 GitHub logo

DEPLOYMENT OPTIONS FOR GITHUB

GitHub has only one possibility of deployment and that is using their Cloud solution and that only they host the company's environment. This part does not meet the requirements for Thales.

The only part that a company can self-manage is the runners for the test/build process.

INTEGRATION POSIBILITIES WITH GITHUB

GitHub offers a sophisticated integration with different applications, GitLab has a repository with applications and detailed description on how to build these integrations.

Besides their own made integrations, they offer a marketplace with plugins and applications for all needs. A Company has the possibility to change their environment to their wishes and requirements. It also makes migrating to and from the platform more accessible.

Categories
API management
Chat
Code quality
Code review
Continuous integration
Dependency management
Deployment
IDEs
Learning
Localization
Mobile
Monitoring
Project management
Publishing
Recently added
Security
Support

Figure 30 GitHub integration

12.5 CLOSER LOOK AT AZURE DEVOPS

Azure DevOps is the third solutions that will be investigated, it is considered a big DevOps vendor, This platform is also from Microsoft and is considered the third biggest DevOps environment. While it is the best for companies that work a lot with Windows environment, a company can manage their whole company using Azure, from Intune as a device manager, active directory to manage rolls and access for their employees.

Azure in general offers a great solution for a company.

Azure offers some of the best application lifecycle management tools on the market.



Figure 31 Azure DevOps logo

DEPLOYMENT OPTIONS FOR AZURE DEVOPS

Azure DevOps has the same deployment options as GitHub, the only option is to let the environment be hosted online by Microsoft itself. While this can be great for most companies in the case for Thales it is not accepted. The pipeline runners can be hosted locally on a server.

INTEGRATION POSIBILITIES WITH AZURE DEVOPS

Azure offers the best IDE integration, with Visual Studio Code being their own application.

Azure DevOps lacks integration possibilities as Azure is more of a local integration system. It does offer integration solutions with GitHub.

12.6 CLOSER LOOK AT LIFT AND SHIFT SOLUTION

The lift and shift solution are the same environment as the environment currently being used. It consists of the main 4 applications that perform the main 4 core features. These are Visual Studio remote Code, Bitbucket, Jenkins and Nexus. Together they perform the following functions:

- **Writing code** for Visual Studio Code.
- **Version control management** is done by Bitbucket.
- **Building** and **testing** code is done by Jenkins.
- **Artifacts** are stored in Nexus.

The goal of a lift and shift migration is to make an exact copy of the current system and place it in a new in this case a Cloud-native platform.

An important step is to make sure that the applications you do this with are Cloud compatible and support Cloud-native features like autoscaling to get the best performance from the migration. With the last few years Cloud-native trend has become more common and vendors update their applications to be able to run on a Cloud-native environment.



Figure 32 Lift and shift methods

DEPLOYMENT OPTIONS FOR LIFT AND SHIFT SOLUTION

The lift and shift solution can be implemented using different solutions. In this case all the 4 applications need to be compatible with the same deployment method to make it as easy to manage as possible. In this case all the applications are compatible with Kubernetes, A normal version option and for Nexus, Bitbucket an online option where the vendors host the environment.

12.7 EVALUATIONS

12.7.1 MAIN AND SUB QUESTIONS EVALUATION

Main question:

- **How could a web-based development environment for TACTICOS applications be implemented on a Cloud-native platform?**
 - o In the advisory, the main question is addressed using all the results and conclusions. In short, a web-based development environment will be an improvement over the current system, due to the Cloud-native platform making the system easier to manage and help save resources.

Sub question:

- **What is a web-based development environment?**
 - o This question is answered during the literature study. In short, a web-based environment stands for an environment where the core functionality and applications can be accessed using a web browser.
- **What is a Cloud-native environment?**
 - o This question is also answered in the literature study, in short, a Cloud-native environment is an environment that consists of Cloud-native techniques, like containers, micro-services
- **What is needed in development environment for TACTICOS applications?**
 - o This question is mostly answered in the requirements/interviews. Thales requires strict security rules on their environment. These are mostly around the servers and that these cannot have any direct connection with the internet.
- **What are the advantages and disadvantages of going Cloud-native?**
 - o This question mostly got answered in the literature study. The main advantages of Cloud-native are Integration, lightweight and scalability. The main disadvantage is taking the step towards a Cloud-native environment which takes a lot of work.
- **What are the requirements for a web-based development environment to be run on a cloud-native platform?**
 - o The requirements that are needed are part of the requirements, the web-based environment does not need a lot to run on a Cloud-native platform. A development environment consists of at least an IDE, a version management system, code building and an artifact storage. Any of the software used to run any of these functions need to be able to run on containers, or other Cloud-native techniques.
- **What web-development tools are available on the market?**
 - o Depending on what method a company wants it can change. In short, a company can choose for an all-in-one solution, lift and shift method and special Cloud-native development applications, however the last is new

since most companies are updating their application to be able to run on Cloud-native infrastructure.

- **How would the design and implementation of a web-based development environment be run on a Cloud-native landscape?**
 - o The full answer can be found in the literature study, it should at least follow these 4 rules. Start this process, set the requirements, setup the cloud environment used to deploy the development environment the last big step before converting is to make sure to port all the code that it can be used in the new environment.

12.7.2 REQUIREMENTS EVALUATION

	ID	Description	MoSCoW	GitLab	Lift and shift
Business requirements	BR-01	A possible container solution should not be tailor made and be commercially available.	Must	Yes	Yes
	BR-02	A design should be able independent. One container solution/program can be replaced with another.	Must	Yes	Yes
	BR-03	The design must be well documented.	Must	Yes	Yes
	BR-04	Employees of these teams must be able to reproduce the proof of concept.	Must	Yes	Yes
	BR-05	The design should not have a vendor lock-in	Must	Maybe	Yes

Table 14 Business requirements

	ID	Description	MoSCoW	GitLab	Lift and shift
User requirements	UR-01	The IDE must be accessible from a browser.	Must	Yes	Web version of VSC can be used not tested
	UR-02	Developers must have access to the following functions in an IDE: <ul style="list-style-type: none"> • Compiler • Debugging tools • Code completion 	Must	Currently being improved	Yes

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

		• Refactor			
	UR-03	Developers have access to plugins for testing	Could	Currently being improved	Yes
	UR-04	Artifacts must be built automatically	Must	Yes	yes
	UR-05	Developers must have access to a version control and merge support.	Must	Yes	Yes
	UR-06	Developers must have access to an artifact storage.	Must	Yes	Yes
	UR-07	The design must support a static code check for example SonarQube and Coverity.	Could	Yes	Yes
	UR-08	The advantages and disadvantages of possible solutions must be described	Must	Yes	Yes
	UR-09	The features of different solutions must be described and documented	Must	Yes	Yes

Table 15 User requirements

	ID	Description	MoSCoW	GitLab	Lift and Shift
System requirements	SR-01	The finished product must run in a local environment with no access to the internet and be managed locally.	Must	Yes	Yes
	SR-02	The system must have high availability.	Should	Yes	Yes
	SR-03	The system should have scalability for at least 500 engineers.	Should	Yes	Yes
	SR-04	The new environment must have auto scaling capabilities.	Could	Yes	Yes

Table 16 System requirements

BUSINESS REQUIREMENTS

BR-01 – Both solutions are off the shelf solutions and available for any company.

BR-02 – While tested on only one solution (Kubernetes) but the software packages also function on other container solutions.

BR-03 – Both designs are well documented solutions, and have a big community supporting it with tutorials.

BR-04 – The guides made in this document will help create the prototype.

BR-05 – For the GitLab solution it can be considered vendor lock-in as it is the only vendor of the whole system, but they do offer great export solutions in case a company is willing to switch. The lift and shift solution use different vendors and has other export solutions.

USER REQUIREMENTS

UR-01 - For GitLab it is easily possible, Visual Studio Code has their own web IDE, it has not been tested but it nearly functions the same.

UR-02 – For the GitLab solutions it is under construction due to a bug found in the newest update (Dec 19, 2022). With Visual Studio Code all these features exist.

UR-03 – Exactly the same answer as UR-02.

UR-04 – Both solutions offer pipeline and focus on CI/CD

UR-05 – Both solutions offer their own version management systems.

UR-06 – Both solutions have their own artifact storage solution.

UR-07 – Both solutions have support on their own for static code checks, the GitLab solution is a bit better, but Jenkins has better plug-in integration to improve it.

UR-08 – The advantages and the disadvantages of both solutions have been well documented.

UR-09 – The features of both solutions are well described in both “Closer look” appendix.

SYSTEM REQUIREMENTS

SR-01 – Both solutions are working in a local environment and both solutions work without internet and only in localhost.

SR-02 – Kubernetes offers extensive list of features to improve high availability, like rolling updates, in case a container gets errors a new one spins up and autoscaling.

SR-03 – Depending on the available resources it can be scaled to as many containers needed.

SR-04 – Both solutions offer auto scaling from Kubernetes, and both solutions offer integration with these functions.

12.7.3 CONDITIONS EVALUATION

- **Must work within a Cloud-native Environment.**
 - o Nearly all popular solutions on the market have the possibility to run on a Cloud-native environment.
- **Solution must work within a Linux based operating system.**
 - o All mentioned solutions work on Linux as Linux is a popular OS for servers.
- **Preferred solution with a Kubernetes orchestrator.**
 - o All solutions shown can and have been run on a Kubernetes environment with a Kubernetes orchestrator.
- **Applications in a solution must be able to work together while running stand alone.**
 - o The GitLab solution does run by itself, but in the lift and shift environment each application used can run by itself and is not dependent on another application.
- **Solution must have an ease of management.**
 - o The core idea of Cloud-native is to make an environment easier to manage.
- **The system should be able to run in a production environment.**
 - o Unfortunately, this has not been tested due to not having access to a production environment.
- **Must integrate with different applications to improve further.**
 - o Each solution that has been tested offers a way to integrate applications in every step of the DevOps cycle.

12.7.4 DESIGN PRINCIPLES EVALUATION

ID	Design principle	Used?	Comment
DP 1	If Thales want a new development environment that makes use of the newest technologies, then Thales should use a Cloud-native development environment, because the Cloud-native trend is relatively new and supported by world leading companies help design the newest techniques in an open-source fashion.	partly	GitLab is using the newest techniques. While the lift and shift solution might not use the newest and greatest techniques in their application.
DP 2	If Thales does not want a vendor lock on their systems, then a Cloud-native solution can help Thales because the CNCF is designed with the idea of multiple applications and micro services	Yes	Both solutions offer a wide variety of vendors that can integrate with their systems.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

	working together and have possibilities to integrate which each other.		
DP 3	If Thales want to save money eventually, then Thales should go for a Cloud-native solution, because Cloud-native solutions are built with the idea of scalability and can auto scale with the demand. This together with the fact that Cloud-native solutions are designed to be more efficient with the resources.	Yes	A Cloud-native solution has the possibility to save resources as it can scale down when the demand is lowering.
DP 4	If Thales want to have a simple way to manage their systems, then a Cloud-native solution can help Thales, because Cloud-native application try their best with an ease of configuration mindset. Most Cloud-native solution have a way to connect another application to theirs with just an API.	Yes	Cloud-native applications are made to be as easy to implement as possible, managing an application is part of it.
DP 5	If Thales want to have multiple ways to implement a new system, then moving to a Cloud-native solution can help Thales because there are at minimum 4 ways to choose from in moving their systems to a Cloud-native environment	Yes	Both solutions offer a wide variety of products and can be setup up completely different with other applications and still do the same job.
DP 6	If Thales want to have an on the shelf solutions, then Thales can consider a Cloud-native solution because in the Cloud-native landscape a lot of different applications exist which Thales can use to create their own solution.	Yes	Cloud-native applications is an active trend in the IT, many new companies and startups are developing applications to make certain tasks automatic.
DP 7	If Thales is looking for an All-in-one solution, then Cloud-native can be a great option because there is great application in the Cloud-native landscape like GitHub and GitLab.	Yes	Applications like GitLab are great all-in-one solutions for any company, the make a great base on which can easily be built to improve further.
DP 8	If Thales needs to have their environment in run locally then Cloud-native can help Thales because Cloud-native does not mean that it is only accessible in the Cloud, most Cloud-native	Yes	While Cloud-native sounds like only an online solution, it is not. Some solutions offer to

	applications can be run locally on an on-premise server.		be run locally. Like both tested solutions.
--	--	--	---

Table 17 Design principle evaluation

12.8 TESTING RESULTS

12.8.1 SETTING UP THE TEST ENVIRONMENT

Setting up the test environment was done with a simple Minikube single node cluster. A both solutions are put in a different namespace inside of the cluster, these splits both solutions from each other and still have the same settings.

For GitLab the first step was to read into deployment solutions, run it directly on Ubuntu or deploying it on Kubernetes. In consultation with the stakeholder “Willy” the choice has been set on the Kubernetes solution. GitLab has a large community and offers installation guides for all their deployment possibilities. One of these ways is Helm, which is a package manager for Kubernetes that allows maintainers extra control and manageability. First Helm has been installed into the environment, which helped setting up the environment. Adding the right repository made sure that with a handful of commands a whole environment consisting of multiple applications can be deployed. In the Helm charts there is the possibility to make changes to the deployment and configuration before deploying, the Helm charts are written in YAML code and editing values inside can change the amount of application setup, scalability and more. With the application deployed a problem came up that a free license is not accepted. So, the deployment did run and was accessible in the browser but the account created with a free license was not accepted. So other tests have been performed using the web version.

The second environment was the lift and shift method, with the goal of moving the four most important applications of the current development environment to a Cloud-native solution. The idea was to move all the applications to Kubernetes as the underlying Cloud-native platform. The next step was to find the best way to deploy the applications. All the options considered it seemed again that Helm was by far the better solution. It gives more control on the deployed application compared to deploying it the default way directly on Kubernetes. So again, Helm has been chosen to deploy the application. Ignoring the warnings on the Atlassian website about needing a license to just try it anyway which made it so that the Bitbucket stack could not deploy. The other solutions gave no problems and just came to life after some long waiting times. The configuration files are stock and no changes have been made in the Helm charts.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

The Visual Studio Code remote development environment asked for an extra application called Docker Desktop, as it would be considered a middle man for Visual Studio to talk to and deploy the containers on the Kubernetes platform.

12.8.2 TESTING THE IDES

The first test was to see if the IDEs function work as intended. This test consists of the basics of writing, running and debugging code. This test also checks if the IDE can be run on a Cloud-native platform.

GITLAB IDE TEST

The IDE is considered one of the few basic functions, it is important to know what GitLab offers, if it can run on a Cloud-native platform and gives the developer the option to perform their work.

The test has been performed using test code.

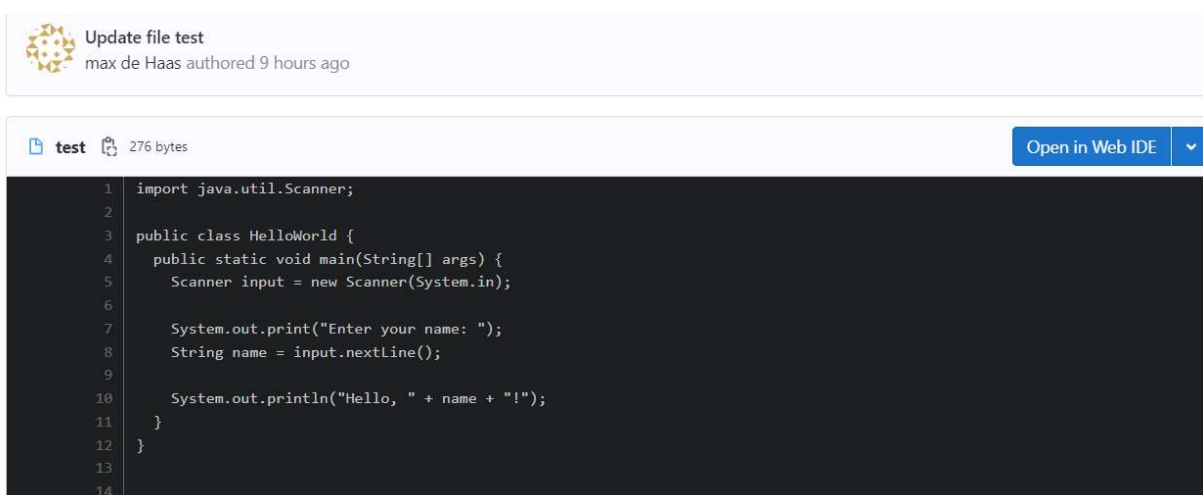


Figure 33 Shows that GitLab has a web IDE

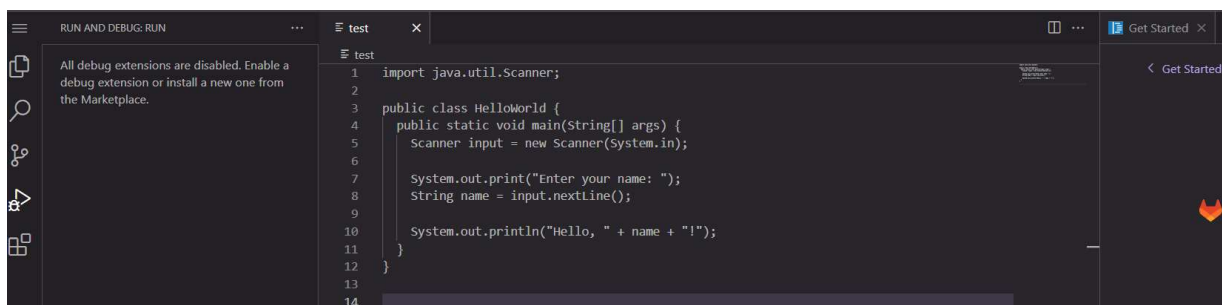


Figure 34 Run/debug option

In the screenshots above it is visible that code can be typed, managed and there is an option to run/debug, this is temporally disabled due to the following message.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

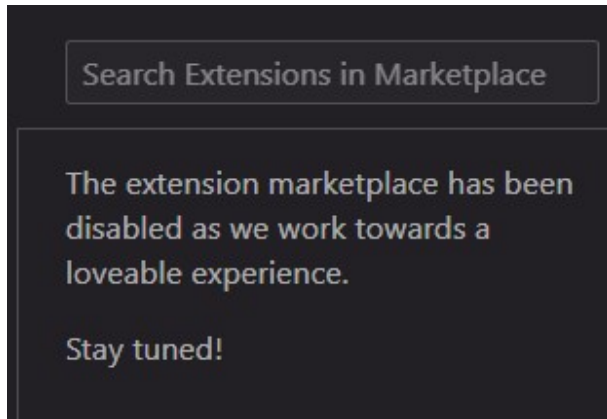


Figure 35 Store warning being closed

While this is the web version of GitLab, with a full license it can be run locally.

NAME	READY	STATUS	RESTARTS	AGE
pod/gitlab-certmanager-7cb7797848-sq2gk	1/1	Running	3 (21m ago)	105m
pod/gitlab-certmanager-cainjector-5968cb88f9-6fnvt	1/1	Running	5 (18m ago)	105m
pod/gitlab-certmanager-webhook-797bcff548-tztlk	1/1	Running	3 (21m ago)	105m
pod/gitlab-gitlab-0	1/1	Running	4 (21m ago)	105m
pod/gitlab-gitlab-exporter-84c54c95dc-fq95g	1/1	Running	3 (21m ago)	105m
pod/gitlab-gitlab-runner-7cfff5c5685-bgkbp	0/1	CrashLoopBackOff	21 (52s ago)	105m
pod/gitlab-gitlab-shell-6c7fcc5859-8jd2j	1/1	Running	3 (21m ago)	105m
pod/gitlab-gitlab-shell-6c7fcc5859-xlflw	1/1	Running	4 (21m ago)	105m
pod/gitlab-issuer-1-vgpdw	0/1	Completed	0	105m
pod/gitlab-kas-785bd6b4c5-k55ch	1/1	Running	3 (21m ago)	105m
pod/gitlab-kas-785bd6b4c5-rnkhn	1/1	Running	3 (21m ago)	105m
pod/gitlab-migrations-1-znmss	0/1	Completed	0	105m
pod/gitlab-minio-74467697bb-pj9cc	1/1	Running	3 (21m ago)	105m
pod/gitlab-minio-create-buckets-1-9gq48	0/1	Completed	0	105m
pod/gitlab-nginx-ingress-controller-748846cd48-6ln6q	1/1	Running	4 (21m ago)	105m
pod/gitlab-nginx-ingress-controller-748846cd48-dpjpx	1/1	Running	3 (21m ago)	105m
pod/gitlab-postgresql-0	2/2	Running	7 (21m ago)	105m
pod/gitlab-prometheus-server-6bf4fffc55-6mjhv	2/2	Running	6 (21m ago)	105m
pod/gitlab-redis-master-0	2/2	Running	6 (21m ago)	105m
pod/gitlab-registry-df8c6797b-dv86v	1/1	Running	3 (21m ago)	105m
pod/gitlab-registry-df8c6797b-f6bfm	1/1	Running	3 (21m ago)	105m
pod/gitlab-sidekiq-all-in-1-v2-5bd767889d-p746p	1/1	Running	3 (21m ago)	105m
pod/gitlab-toolbox-5546697c56-8sdrx	1/1	Running	3 (21m ago)	105m
pod/gitlab-websevice-default-56669dfc98-mt5xt	2/2	Running	7 (21m ago)	105m
pod/gitlab-websevice-default-56669dfc98-swxff	2/2	Running	6 (21m ago)	105m

Figure 36 GitLab instances running

As shown in the pictures GitLab offers a great IDE solution based Visual Studio Code.

VISUAL STUDIO CODE IDE TEST

Visual Studio Code is an IDE on its own so knowing that it functions like an IDE. So, the key is that it can be done remotely. To set it up it requires Docker desktop downloaded and that is it. Furthermore, the next step is to install the following plugin:

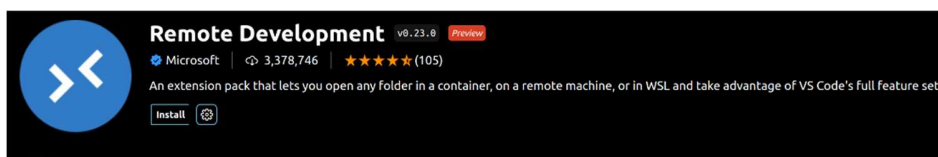
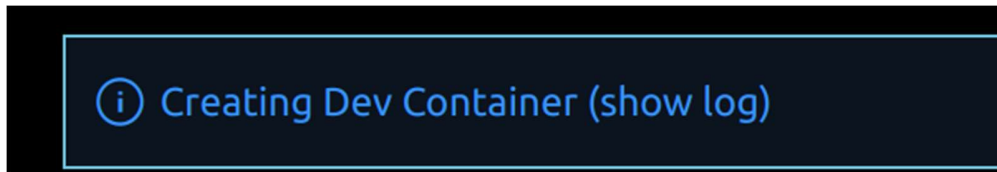


Figure 37 Remote development plugin

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

After this is installed that is it, run the code and this will pop up.



The logs will show the following, while this might not be the fastest method of running code, it will take the load of the Citrix environment and the

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(6/24) Installing libgomp (11.2.1_git20220219-r2)
(7/24) Installing libatomic (11.2.1_git20220219-r2)
(8/24) Installing isl22 (0.22-r0)
(9/24) Installing mpfr4 (4.1.0-r0)
(10/24) Installing mpc1 (1.2.1-r0)
(11/24) Installing gcc (11.2.1_git20220219-r2)
(12/24) Installing musl-dev (1.2.3-r2)
(13/24) Installing libc-dev (0.7.2-r3)
(14/24) Installing g++ (11.2.1_git20220219-r2)
(15/24) Installing make (4.3-r0)
(16/24) Installing c-ares (1.18.1-r0)
(17/24) Installing icu-data-en (71.1-r2)
Executing icu-data-en-71.1-r2.post-install

* If you need ICU with non-English locales and legacy charset support, install
* package icu-data-full.
*

(18/24) Installing icu-libs (71.1-r2)
(19/24) Installing nodejs (16.17.1-r0)
(20/24) Installing npm (8.10.0-r0)
(21/24) Installing mpdecimal (2.5.1-r1)
(22/24) Installing python3 (3.10.9-r0)
(23/24) Installing gcc-zsh-completion (5.8.1-r4)
(24/24) Installing docker-zsh-completion (20.10.20-r0)
Executing busybox-1.35.0-r17.trigger
OK: 508 MiB in 140 packages
Removing intermediate container 99affdad6dfc
--> e81de3468232
Step 3/4 : RUN cd && npm i node-pty
--> Running in 6c218d7e841b

added 2 packages, and audited 3 packages in 4s

found 0 vulnerabilities
Removing intermediate container 6c218d7e841b
--> c965899c7992
Step 4/4 : COPY .vscode-remote-containers /root/.vscode-remote-containers
--> b0fe07b00ba5
Successfully built b0fe07b00ba5
Successfully tagged vsc-volume-bootstrap:latest
```

Figure 38 Container run code

12.8.3 VERSION MANAGEMENT SYSTEM

The second test is the version management system, this system stores the code and shows the latest changes to code. This test in GitLab is done with editing code using the web editor and committing the code. For the Bitbucket environment it is important to see if it can run on a Cloud-native environment.

GITLAB VERSION CONTROL SYSTEM

The following picture shows a new commit (commit 37f756) which is a change from the previous code. The new commit has 9 additions and 1 line has been deleted.

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

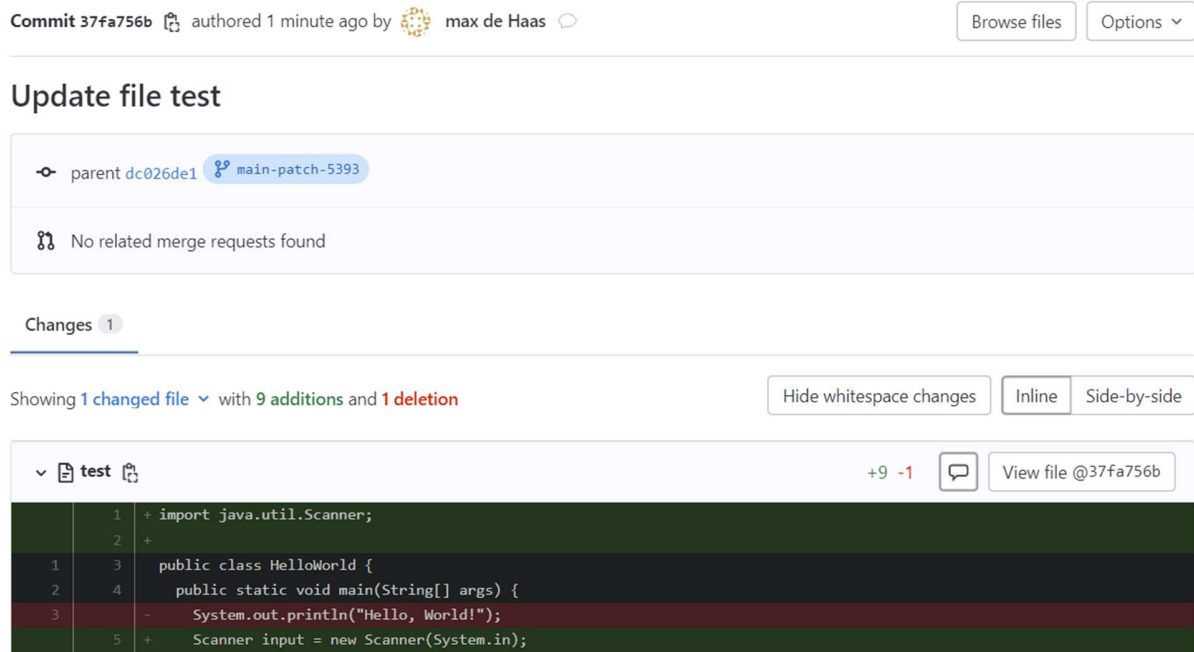
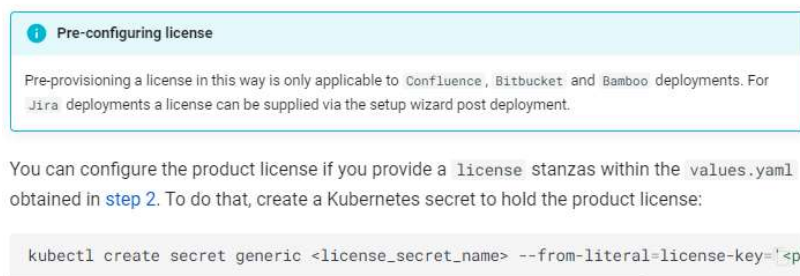


Figure 39 GitLab version control system

BITBUCKET VERSION CONTROL MANAGEMENT

Bitbucket will also be deployed using Helm, there is a warning that you need a license, see step 7:

7. Configure license



Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

GitLab offers a test pipeline to show you what it does and what code building is. Just running this test shows that there is an option to build a pipeline that works.

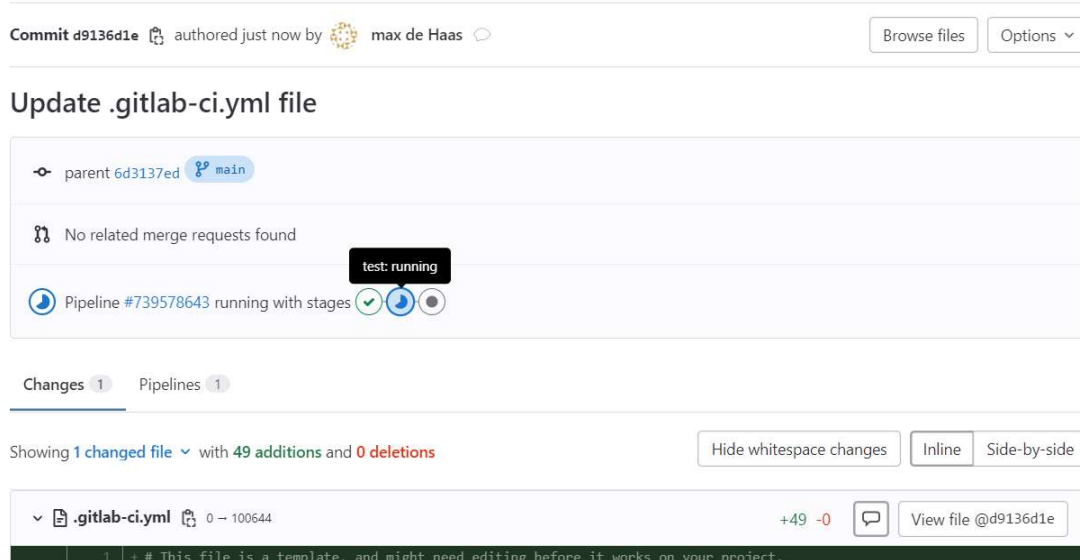


Figure 41 GitLab pipeline

The pipeline builds/tests and deploys the code as it is supposed to do.

JENKINS PIPELINE TOOL

To test Jenkins, it has been installed using Helm. Jenkins is an open-source tool with a big community. So, there are up to date GitLab Charts for Kubernetes. These charts contain all the settings to run and start up Jenkins on a Kubernetes platform. Making sure that it works.

```
^Cmaintainer@maintainer-HP-ZBook-17-G4:~$ kubectl get all
NAME                READY   STATUS    RESTARTS   AGE
pod/jenkins-0       2/2     Running   0           9m39s

NAME                TYPE          CLUSTER-IP      EXTERNAL-IP  PORT(S)          AGE
service/jenkins     ClusterIP     10.102.163.185  <none>       8080/TCP         9m39s
service/jenkins-agent ClusterIP     10.105.5.37    <none>       50000/TCP        9m39s
service/kubernetes   ClusterIP     10.96.0.1      <none>       443/TCP          8d

NAME                READY   AGE
statefulset.apps/jenkins 1/1     9m39s
```

Figure 42 Jenkins pods running

Thesis – UNCLASSIFIED- Web-based development environment for TACTICOS applications implemented on a cloud-native platform.

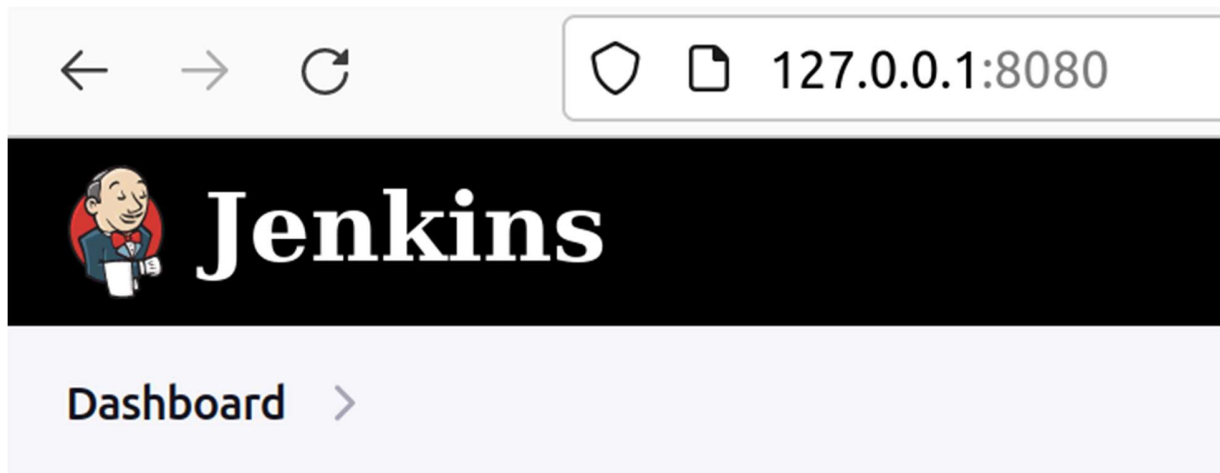


Figure 43 Jenkins main page

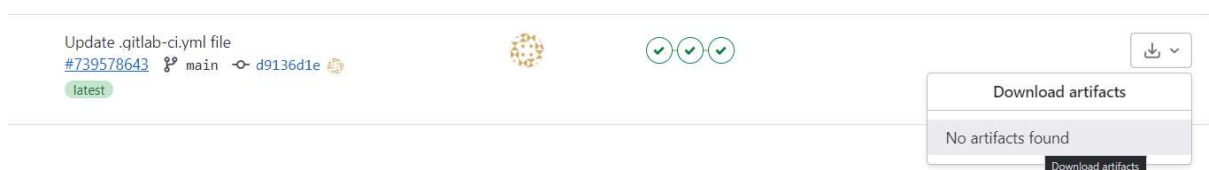
In the pictures above, it is shown that Jenkins can run on a Kubernetes environment.

12.8.5 ARTIFACT STORAGE TEST

The artifact storage is the final needed function, here are the artifacts stored after they are built and tested. The artifacts can be used in newer code or be used as a guideline for new code.

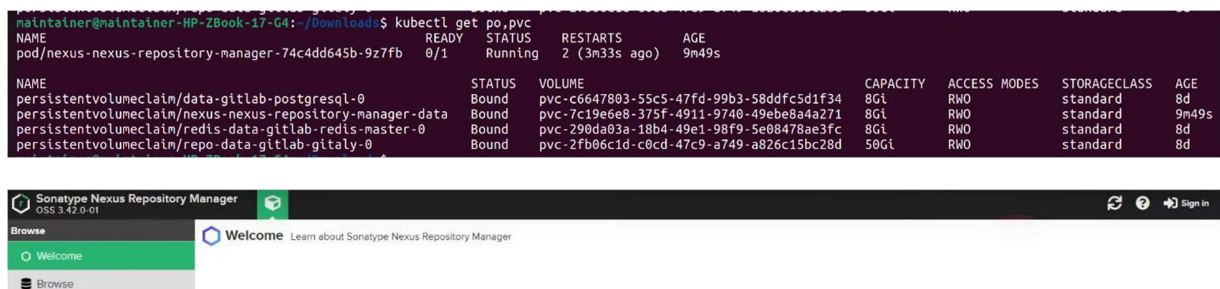
ARTIFACT STORAGE IN GITLAB

The artifact storage in GitLab was not able to be tested. Due to it being a free trail and the test code not giving any artifacts is the only thing that can be done is showing where the artifacts were supposed to be.



ARTIFACT STORAGE IN NEXUS

So, the goal is to move the Nexus on a Cloud-native platform



12.9 GUIDES AND MANUALS

12.9.1 KUBERNETES INSTALLATION

SETUP DOCKER

Setup a fresh Ubuntu install to start with root permissions. Also make sure the environment has the latest updates.

Get all the updates for the **apt** package index

- **sudo apt-get update**

Install ca-certificates (necessary for docker), Curl, gnupg and lsb-release.

- **sudo apt-get install **
**ca-certificates **
**curl **
**gnupg **
lsb-release

Next step is adding the Docker's GPG key.

- **sudo mkdir -p /etc/apt/keyrings**
- **curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg**

The following command can be used to setup a repository.

- **echo **
"deb [arch=\$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu

\$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null

Now update the **apt** package index again.

- **sudo apt-get update**

The final step is to download docker using the following command

- **sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin**

SETUP MINIKUBE

Start with making Docker the default driver.

- **minikube config set driver docker**

Download and install Minikube using the following commands.

- **curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64**
- **sudo install minikube-linux-amd64 /usr/local/bin/minikube**

Minikube is installed and now start it using

- **Minikube start**

SETUP NAMESPACES

The following commands to create namespaces and separate both environments from each other. Both commands will create a different namespace so the command must be performed once for both environments.

- **kubectl create namespace GitLab**
- **kubectl create namespace LaS**

To switch between both namespaces the following command can be used.

- **kubectl config set-context --current --namespace=<name>**

SETUP HELM

Helm is easy to setup and important for quick application setup as a package manager for Kubernetes

Installing and setting up Helm can be done with the following commands.

- **curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null**
- **sudo apt-get install apt-transport-https --yes**
- **echo "deb [arch=\$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list**
- **sudo apt-get update**
- **sudo apt-get install helm**

12.9.2 SETTING UP THE GITLAB ENVIRONMENT

The following commands are used to setup a default GitLab solution for tests.

```
helm repo adds gitlab https://charts.gitlab.io/  
helm repo update  
helm upgrade --install gitlab gitlab/gitlab \  
  --timeout 600s \  
  --set global.hosts.domain=example.com \  
  --set global.hosts.externalIP=10.10.10.10 \  
  --set certmanager-issuer.email=me@example.com \  
  --set postgresql.image.tag=13.6.0
```

Now nothing more must be done except wait for all the containers to spin up.

12.9.3 SETTING UP THE LIFT AND SHIFT SOLUTION ENVIRONMENT

After all the previous steps have been completed except for the GitLab environment setup then the following steps can be performed to setup the lift and shift solution.

To start install the application Visual Studio Code from the official website.

Same goes for Docker Desktop and add your user to the Docker group. (Possibly already done in during the previous setup)

- **sudo usermod -aG docker \$USER**

This step connects Docker to the account used and recognizes Minikube to run containers.

Inside of Visual Studio Code go to marketplace and install the Remote development environment. It gives the user the option to select open in container in the bottom left. (This can be done with a server in a different place but in the test environment there is only 1 place)

INSTALL JENKINS

Jenkins can be installed using Helm with 3 simple commands.

- **helm repo add jenkins <https://charts.bitnami.com/bitnami>**
- **Helm repo update**
- **Helm install jenkins jenkins/jenkins**

The first command adds the repository where the correct Helm charts can be found.

The second command updates the repository list in Helm.

The third commands pull and deploys the code from the repository.

INSTALL NEXUS REPOSITORY

To start installing Nexus repository the repository must be added to the repository list. This can be done using the following command:

- **Helm repo add sonatype <https://sonatype.github.io/helm3-charts/>**

Update the repository list using:

- **Helm repo update**

The following command can be used to deploy Nexus:

- **Helm install nexus-rm sonatype/nexus-repository-manager**

This automatically pulls the latest version from the repository and deploys it.

13 REFERENCES

- Amazon. (2022). *Microservices*. Retrieved from Amazon: <https://aws.amazon.com/microservices/>
- Atlassian. (2022). *What is version control?* Retrieved from Atlassian: <https://www.atlassian.com/git/tutorials/what-is-version-control>
- Cloud Native Computing Foundation. (2022). *CNCF Cloud Native Interactive Landscape*. Retrieved from Cloud native landscape: <https://landscape.cncf.io/>
- Community of Wikipedia. (2022, 10 17). *Integrated development environment*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Integrated_development_environment
- community on Wikipedia. (2022, 06 16). *Software build*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Software_build
- Docker. (2022). *Install Docker Engine on Ubuntu*. Retrieved from Docker docs: <https://docs.docker.com/engine/install/ubuntu/>
- Easy, D. M. (Director). (2022). *Deploy Nexus Repository on DigitalOcean using Helm Charts* [Motion Picture].
- G2. (2022). *Top 10 Git Alternatives & Competitors*. Retrieved from G2: <https://www.g2.com/products/git/competitors/alternatives>
- Gartner. (2022). *JIRA Software vs Azure Boards vs GitHub Enterprise vs GitLab*. Retrieved from Gartner: <https://www.gartner.com/reviews/market/enterprise-agile-planning-tools/compare/product/atlassian-software-vs-azure-boards-vs-github-enterprise-vs-gitlab>
- Git. (2022). *Branching and Merging*. Retrieved from Git: <https://git-scm.com/about/branching-and-merging>
- GitHub. (2022). *Actions*. Retrieved from GitHub: <https://github.com/features/actions>
- GitHub. (2022, 05 23). *DevSecOps explained*. Retrieved from GitHub: <https://resources.github.com/devops/fundamentals/devsecops/>
- GitHub. (2022). *Features*. Retrieved from GitHub: <https://github.com/features>
- GitHub. (2022). *GitHub integrations*. Retrieved from GitHub: <https://github.com/integrations>
- GitHub. (2022). *Marketplace*. Retrieved from GitHub: <https://github.com/marketplace>
- GitHub. (2022). *Pricing*. Retrieved from GitHub: <https://github.com/pricing>

GitLab (Director). (2021). *DevOps Platform - GitLab Product Overview Demo* [Motion Picture].

GitLab. (2022). *Azure DevOps vs GitLab*. Retrieved from GitLab:
<https://about.gitlab.com/devops-tools/azure-devops-vs-gitlab/>

GitLab. (2022). *Deploy the GitLab Helm Chart*. Retrieved from GitLab:
<https://docs.gitlab.com/charts/installation/deployment.html>

GitLab. (2022). *DevOps Tools Landscape*. Retrieved from GitLab:
<https://about.gitlab.com/devops-tools/>

GitLab. (2022). *GitLab Integrations*. Retrieved from GitLab:
<https://docs.gitlab.com/ee/integration/>

GitLab. (2022). *GitLab vs GitHub*. Retrieved from GitLab:
<https://about.gitlab.com/devops-tools/github-vs-gitlab/>

GitLab. (2022). *Learn GitLab with tutorials*. Retrieved from GitLab:
<https://docs.gitlab.com/ee/tutorials/>

GitLab. (2022). *Pricing*. Retrieved from GitLab: <https://about.gitlab.com/pricing/>

GitPod. (2022). *Pricing*. Retrieved from GitPod: <https://www.gitpod.io/pricing>

Helm. (2022). *Installing Helm*. Retrieved from Helm: <https://helm.sh/docs/intro/install/>

Hull, B. (2020). *What is the difference between GitHub and Artifactory?* Retrieved from Quora: <https://www.quora.com/What-is-the-difference-between-GitHub-and-Artifactory>

Jenkins. (2022). *Kubernetes*. Retrieved from Jenkins:
<https://www.jenkins.io/doc/book/installing/kubernetes/>

Jetbrains. (2022). *Remote development*. Retrieved from JetBrains:
<https://www.jetbrains.com/remote-development/#why-go-remote>

katalon. (2022). *Best 14 CI/CD Tools You Must Know*. Retrieved from Katalon:
<https://katalon.com/resources-center/blog/ci-cd-tools>

Linthicum, D. (2015, 06 16). *The pros and cons of going cloud-native*. Retrieved from InfoWorld: <https://www.infoworld.com/article/2935616/the-pros-and-cons-of-going-cloud-native.html>

Marcin Aman, M. W. (2022, 09 29). *Pros and Cons of Cloud IDE*. Retrieved from virtuslab:
<https://virtuslab.com/blog/pros-and-cons-of-cloud-ide/>

Mind tools content team. (2022). *Mindtools*. Retrieved from Stakeholder analysis: https://www.mindtools.com/pages/article/newPPM_07.htm

Minikube. (2022, 12 29). *Get Started!* Retrieved from Minikube:
<https://minikube.sigs.k8s.io/docs/start/>

odalovic. (2020, 03 04). *The clash of the giants - IntelliJ Ultimate vs Visual Studio Code*. Retrieved from odalovic: <https://www.odalovic.com/blog/intellij-ultimate-vs-visual-studio-code-battle>

Opengroup. (2022). *The TOGAF® Standard, 10th Edition*. Retrieved from Opengroup: <https://www.opengroup.org/togaf>

Oracle. (2022). *Wat is cloud-native?* Retrieved from Oracle:
<https://www.oracle.com/nl/cloud/cloud-native/what-is-cloud-native/>

Prometheus. (2022). *Prometheus*. Retrieved from Prometheus: <https://prometheus.io/>

Sahid. (2022, 08 5). *Kubernetes Namespace: Everything you need to know*. Retrieved from K21 academy: <https://k21academy.com/docker-kubernetes/kubernetes-namespace/#:~:text=Working%20with%20Kubernetes%20Namespaces,-Create%20a%20new&text=Because%20this%20can%20be%20time,%E2%80%93current%20%E2%80%93namespace%3DK21>

Sonatype. (2021). *nexus vs artifactory*. Retrieved from sonatype: <https://info.sonatype.com/nexus-vs-artifactory/>

Sysdig. (2022). *What is Kubernetes CrashLoopBackOff?* Retrieved from Sysdig: <https://sysdig.com/blog/debug-kubernetes-crashloopbackoff/>

Telerik. (2022). *Pricing*. Retrieved from Telerik:
<https://www.telerik.com/purchase.aspx?filter=web>

Verites. (2022). DevOps Services. Retrieved from Verites:
<https://www.veritis.com/solutions/devops/>

VMware. (2022). *Containers*. Retrieved from VMware:
<https://tanzu.vmware.com/containers>

VMware. (2022). *What are Microservices?* Retrieved from VMware:
<https://www.vmware.com/topics/glossary/content/microservices.html>

Wikipedia. (2022, 12 13). *List of mergers and acquisitions by Microsoft*. Retrieved from Wikipedia:
https://en.wikipedia.org/wiki/List_of_mergers_and_acquisitions_by_Microsoft

14 TABLE OF FIGURES AND TABLES

Figure 1 Thales Nederland headquarters (GIB)	7
Figure 2 The TOGAF standard (The Open Group).....	11
Figure 3 Cloud-native trend (Oracle)	15
Figure 4 Conceptual model	16
Figure 5 power/interest grid example	21
Figure 6 Power/Interest Grid	22
Figure 7 Development environment drawn	30
Figure 8 Cloud-native landscape (CNCF)	31
Figure 9 Cloud-native trend (Oracle)	32
Figure 10 Containers.....	32
Figure 11 Microservices	32
Figure 12 DevOps drawing.....	33
Figure 15 What is GIT.....	36
Figure 16 Cloud-native computing foundation CI/CD.....	37
Figure 15 Current situation maintainer	45
Figure 16 Future situation maintainer	45
Figure 17 Current situation Developer	46
Figure 18 Future situation with GitLab developer.....	46
Figure 19 New development environment test environment	49
Figure 20 Architecture drawing current developer	64
Figure 21 Architecture drawing current maintainer	65
Figure 22 Future development environment GitLab.....	66
Figure 23 Future maintainer environment GitLab	67
Figure 24 Future Development environment Kubernetes	68
Figure 25 Future Maintainer environment Kubernetes.....	69
Figure 26 GitLab logo	70
Figure 27 GitLab features (GitLab, 2023)	70
Figure 28 Applications integration GitLab	72
Figure 29 GitHub logo.....	73
Figure 30 GitHub integration.....	74
Figure 31 Azure DevOps logo	75

Figure 32 Lift and shift methods	76
Figure 33 Shows that GitLab has a web IDE	84
Figure 34 Run/debug option.....	84
Figure 35 Store warning being closed.....	85
Figure 36 GitLab instances running	85
Figure 37 Remote development plugin	85
Figure 38 Container run code.....	86
Figure 39 GitLab version control system.....	87
Figure 40 license key message	87
Figure 41 GitLab pipeline	88
Figure 42 Jenkins pods running	88
Figure 43 Jenkins main page	89
Table 1 Version management	1
Table 2 Contact information supervisors and graduate	1
Table 3 Glossary and abbreviations.....	5
Table 4 MoSCoW table	9
Table 5 Interviewees and goals.....	10
Table 6 Sub question descriptions and coverage	18
Table 7 Stakeholders and their function	20
Table 8 Sort primary/secondary stakeholder	20
Table 9 Stakeholder's categorization	22
Table 10 Business requirements.....	23
Table 11 user requirements	24
Table 12 System requirements	24
Table 13 Comparison between solutions.....	43
Table 14 Business requirements.....	78
Table 15 User requirements	79
Table 16 System requirements	79
Table 17 Design principle evaluation.....	83