MASTER'S THESIS 2022

Handling Multiple Cloud Service Providers – Common Challenges and Best Practices

Pontus Jaensson, Oscar Wiklund

ISSN 1650-2884
LU-CS-EX: 2022-32

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2022-32

Handling Multiple Cloud Service Providers – Common Challenges and Best Practices

Hantering av flera molntjänstleverantörer – vanliga utmaningar och bästa praxis

Pontus Jaensson, Oscar Wiklund

Handling Multiple Cloud Service Providers – Common Challenges and Best Practices

Pontus Jaensson
po5265ja-s@student.lu.se

Oscar Wiklund
mas15owi@student.lu.se

June 15, 2022

Master's thesis work carried out at IKEA IT AB.

Supervisors: Habib Un Nabi Hillol – habib.hillol@ingka.ikea.com
Lars Bendix – lars.bendix@cs.lth.se

Examiner: Per Andersson – per.andersson@cs.lth.se

Abstract

Customers have a high expectation on both availability and low latency for the applications they are using. In order to fulfill these requirements, organizations can use a Cloud Service Provider (CSP) to deploy their applications in different locations within close proximity to their customers. However, all CSPs are not available everywhere in the world which requires global organizations to use multiple CSPs to reach their customers and satisfy their expectations.

The use of multiple CSPs comes with challenges that need to be identified. In order to reveal all these challenges both an extensive literature study and interviews with developers at IKEA were conducted. Several challenges were revealed which in turn were categorized into either a technical or a non-technical challenge. The main technical challenge was found to be portability, i.e. how to move data and applications between CSPs. Other technical challenges that are acknowledged are interoperability, vendor lock-in and security aspects. The non-technical challenges were identified as legal aspects, sustainability, economy and talent management.

Due to the scope of the thesis, the main objective was to identify and explore the best practices for addressing the technical challenges. In order to achieve this, we conducted a literature study intended to identify possible challenges before we explored them further in a practical context through hands-on experiments. This allowed us to conclude that abstractions and variant segregation among others are suitable approaches for addressing the challenges.

Keywords: Abstractions, Cloud Agnostic, Multi-cloud, Portability, Segregated Variants

Acknowledgements

To begin with, we would like to express our gratitude to our academic supervisor Lars Bendix for his valuable support and continuous feedback throughout the entire thesis. His expertise, great humor and unrivaled methods have made all phases of the thesis highly educational and enjoyable from the very beginning to the end. We would also like to thank the entire customer web team at IKEA IT AB for their warm welcoming and support from the very start. It has been a pleasure working with them. Last but definitely not least, we would like to especially thank our company supervisor Habib Un Nabi Hillol for everything he has done for us. His guidance and expertise of cloud computing in combination with successfully introducing us to everything we needed to know about the customer web team and how they work has been invaluable.

Contents

List of Abbreviations

Alicloud – Alibaba Cloud
Azure – Microsoft Azure
CROF – Cloud Resource Orchestration Framework
CSP – Cloud Service Provider
FaaS – Function as a Service
GCP – Google Cloud Platform
IaC – Infrastructure as Code
OS – Operating System
RQx – Research Question number *x*
SPL – Software Product Line
VM – Virtual Machine

Chapter 1
Introduction

During the last two decades the cloud computing paradigm has risen in popularity and has become an approach that almost all organizations use to some extent in their everyday businesses. Today there exists a range of various Cloud Service Providers (CSPs) that offer a wide variety of services on both a local and a more global scale. The use of a single CSP is often sufficient enough to fulfill the requirements of the customers but there also exist scenarios where this is not the case. For example, the fact that not even the largest CSPs are available everywhere in the world results in the need of a multi-cloud approach for an organization interested in exploiting business opportunities in new geographical markets across the globe. Given that the area of cloud computing in general and especially multi-cloud computing is evolving rapidly, both new challenges and practices are constantly introduced. As a consequence, organizations are experiencing difficulties handling multiple CSPs and need an efficient manner to address the current common challenges.

One organization where the use of a single CSP is not enough is IKEA, the organization where this thesis has been conducted. The customer web team within the customer engagement team at IKEA IT AB, hereafter only referred to as the IKEA team, is responsible for a number of software solutions used by different IKEA markets all over the world. However, there are still some markets where the applications are not yet available. Reaching and establishing the applications in new markets is a continuous process for IKEA which in some cases introduces new problems such as the preferred CSP not being available in the specific market region. Hence, the IKEA team has discovered the need for a multi-cloud approach in order to make their products available in all markets they are present in.

Global organizations like IKEA, are also heavily dependent on the availability of their services. Downtimes on their services means that potential customers can not purchase their products which is translated into huge financial losses for the organization. A scenario that the recent 2021 downtime of Facebook illustrated, where a few hours of outage resulted in a multi-million dollar loss in revenue for the company. Adopting a multi-cloud approach

could increase the availability of services by protecting the IKEA team from downtimes on a specific CSP. Another organizational motivation for adopting a multi-cloud strategy is to avoid being locked to the services of a specific CSP. If applications are prepared to be run on multiple CSPs, changes in pricings or services of a specific CSP are more easily handled and cause less harm if they were to occur.

The IKEA team is almost exclusively working with a single CSP, Google Cloud Platform (GCP). However, since they have discovered the need for more than one CSP for reaching their planned markets with their applications, they have begun exploring other CSPs available in these regions such as the Chinese CSP Alibaba Cloud (Alicloud). They have realized that this new multi-cloud strategy might introduce new uncertainties and unfamiliar challenges which have to be further investigated. Hence, this thesis will thoroughly examine these multi-cloud uncertainties and how they will affect areas such as development, deployment and infrastructure if a multi-cloud approach is adopted. Looking one step further, once the challenges have been identified, there will also emerge a need for handling them efficiently. Therefore, the objectives of this thesis are to identify the challenges and the solutions as well as finding the best practices for handling multiple CSPs. The objectives have resulted in the research questions presented in the following section.

1.1 Research Questions

The thesis will aim to investigate and explore the following research questions:

- **RQ1** – What are the challenges associated with handling multiple CSPs?
- **RQ2a** – What are the best practices for addressing the challenges identified in RQ1 with respect to infrastructure, deployment and development?
- **RQ2b** – Can a variant perspective result in a suitable solution for addressing the challenges identified in RQ1?
- **RQ3** – Can a proof of concept with the findings from RQ2 be implemented?

With respect to the research questions, the thesis can be divided into three separate phases; the *identification of challenges*-phase, the *identification of solutions*-phase and the *practical exploration of the solutions*-phase. The *identification of challenges*-phase will aim to answer RQ1 by studying previous work and established knowledge on the subject as well as conducting interviews with developers at the IKEA team. Once the challenges have been identified, the *identification of solutions*-phase will begin by finding and discussing different practices for handling them. This phase will also include the exploration of RQ2b, if a variant perspective can be used for addressing the challenges from the previous phase. The final *practical exploration of the solutions*-phase will then use and further explore the results from the previous literature studies and a small proof of concept will be implemented.

1.2 Thesis Disposition

The next chapter will introduce some necessary theory that will be used throughout the thesis. It will also describe the context of which the thesis was conducted within as well as the methodology used for each phase. Thereafter, the *Identification of Challenges* chapter will investigate the challenges associated with multiple CSPs before the *Identification of Solutions* chapter will explore possible solutions for handling these challenges. This is followed by the *Practical Exploration of the Solutions* chapter where the findings from the previous chapters are evaluated. The *Discussion and Related Work* chapter will then discuss and reflect upon our work process, the validity concerns of the thesis and the generalizability of the results. This chapter will also discuss existing related work and how the thesis subject could be further investigated and explored by discussing our ideas for future work. Finally, in the *Conclusions* chapter the research questions will be answered and our final recommendations will be given.

Chapter 2
Background

The purpose of this chapter is to establish an understanding of both the context and the important theories of which the thesis is based upon. It is important to both understand how the thesis has been conducted and how the context has influenced the work and the result of the thesis. Secondly, it also clarifies central concepts and theories that the remaining chapters of the thesis are using as well as it assures that no unambiguous perception of the theory is present in the mind of the reader. This is achieved by first introducing IKEA in general and the team where the thesis was conducted followed by a description of the initiating problem and its context. Furthermore, the motivation for why a multi-cloud approach is of interest will be explained. This is followed by a section discussing how we plan to conduct the thesis work as well as our motivations for why we consider it a suitable approach. The chapter is then concluded by introducing the theoretical foundation that is required for understanding the remainder of the thesis.

2.1 Context

The purpose of this section is to introduce the context of IKEA and the initiating problem. Since the context influences the direction and scope of the thesis it is therefore important to describe it in detail. This will also provide the reader with the opportunity to evaluate how their own situation aligns with the investigated domain in order to conclude the relevance of the thesis work. Furthermore, the initiating problem will be analyzed and formulated into research questions that the thesis will address.

2.1.1 IKEA

IKEA is a global furniture company with more than 225 000 employees and present in 63 different countries worldwide [1]. The company was originally founded in 1943 by Ingvar Kamprad and has gone from being a small local business to one of the most well-known

brands in the world.

An important aspect to bear in mind throughout the thesis is that due to the fact that IKEA is a multinational company with a complex corporate structure, it is outside the scope of the thesis to take the whole organization into consideration. Instead, the thesis will be conducted with the customer web team as stated in the introduction chapter. The IKEA team is working with a range of different products used in various markets across the globe. The general theme of the products they are developing is the ability to engage and interact with customers by different solutions hosted in warehouses or online.

Current Practices and Problem Statement

The applications and products that the IKEA team are responsible for are hosted in the cloud. This allows the applications to be deployed close to the customers without forcing IKEA into investing in and maintaining different server centers located in all the markets IKEA is present in. Google Cloud Platform (GCP) is the IKEA team's preferred CSP and they are exclusively working with it even though some initial research of working with other CSPs is being carried out. This is due to the team mostly working with markets in Europe and other places where GCP is available. However, they have recently discovered the need for their applications to be hosted in other markets than the ones they are currently established in, markets like China where GCP is not allowed [2]. This means that the IKEA team is forced to find a way of working with another available CSP in these regions. Working with more than one CSP is an unexplored area for the IKEA team and they are not certain about what impacts this might have on their way of working and current applications which implies that this has to be further investigated.

Another motivation for why the IKEA team is interested in investigating a multi-cloud strategy is to ensure the uptime of their applications. The IKEA team's applications are hosted in cloud environments meaning that they are heavily dependent on the uptime of the CSP's services. An outage at the CSP results in downtime for the IKEA team's applications which directly translates to a revenue loss for IKEA. An example of the impact such a scenario would have is illustrated by the 2021 downtime of Facebook where a five hour outage resulted in an estimated loss in revenue of $65 million for the company, apart from all the negative publicity such an event would cause [3]. There are also examples of downtimes of specific CSPs that have resulted in huge financial losses for not only the CSPs but also for their customers [4]. The IKEA team is therefore looking into a multi-cloud strategy with the motivation that such a strategy could be used to avoid downtimes of a specific CSP by quickly relocating applications to another CSP and hence ensuring the uptime if an incident like this occurs.

Adopting a multi-cloud approach will also prevent the IKEA team from being vendor-locked to a specific CSP, a situation where the cost of switching to another CSP is too high. Currently the IKEA team is very dependent on GCP and their services and are therefore not only vulnerable to downtimes but also to changes of services and price hikes. By avoiding vendor lock-in, the IKEA team could be more flexible in their choice of CSP and for example be able to address the problem related to ensuring uptime for their services more easily. They could also better utilize the strengths of each CSP by being able to switch to another CSP

that offers a better suited solution in terms of performance or price than their current one.

The challenges discussed above illustrate why a multi-cloud adoption is in the greatest interest of the IKEA team. However, as explained, the team lacks experience regarding working with another CSP than GCP. Some of the developers have previously worked with other CSPs but not in a multi-cloud context. Therefore, the purpose of this thesis is to investigate how a multi-cloud strategy could solve the problems presented above and how it would affect the IKEA team in aspects such as development, deployment and infrastructure. The expected result of the thesis is to provide the IKEA team with recommendations for facilitating the process of adopting a multi-cloud strategy. In order to accomplish this, we first need to explore and investigate the challenges associated with a transition to a multi-cloud strategy which will be revealed by answering the following research question.

- **RQ1** – What are the challenges associated with handling multiple CSPs?

Once the challenges have been identified, we would like to explore suitable ways of handling each of them with respect to the context of the IKEA team. Based on our own experiences from the *Configuration management* course at Lund University, our hypothesis is that some kind of variant management concept will most likely be a part of a potential solution for handling multiple CSPs [5]. We have the perception that variations will inevitably occur and we will investigate whether concepts from configuration management can lead to strategies to manage them. We have therefore defined the following two research questions which aim to explore potential solutions to the challenges found in RQ1 and take the previously discussed reasoning into consideration.

- **RQ2a** – What are the best practices for addressing the challenges identified in RQ1 with respect to infrastructure, deployment and development?
- **RQ2b** – Can a variant perspective result in a suitable solution for addressing the challenges identified in RQ1?

Once potential solutions have been identified and then further explored, we would like to investigate if we can implement a proof of concept to realize our findings. The result from and the experience gained by implementing a proof of concept will be used to strengthen our final recommendations that we will provide to the IKEA team regarding how to handle multiple CSPs. This is planned to be explored in the final research question presented below.

- **RQ3** – Can a proof of concept with the findings from RQ2 be implemented?

2.2 Methodology

The purpose of this section is to discuss possible approaches for how to conduct the thesis work. This includes a description of steps planned to be taken and the motivations and reasonings behind them in order for the reader to understand how the research questions are approached. An additional purpose of this section is to provide transparency of the methodology, which enables reproducibility and the ability to validate the findings of the thesis.

The initial thesis work structure is presented in figure 2.1 below. As illustrated, our plan is to begin by researching RQ1 during the *identification of challenges*-phase. This is a two-step process where a literature study about the subject will serve as a broad foundation, targeting the challenges mentioned in the existing literature and research. The literature study will then be complemented by interviews with developers at the IKEA team in order to conclude which specific challenges are most relevant for the context of the thesis. We will then study and analyze the findings from the literature study and the interviews in order to compile them to a final result for RQ1. Our findings will eventually be used as the basis for the remaining research questions. In the *identification of solutions*-phase we will investigate RQ2a and RQ2b in parallel by conducting a second literature study. Once the potential solutions have been identified, RQ2a and RQ2b will be further investigated by exploring the solutions in the *practical exploration of the solutions*-phase. This phase will consist of hands-on experiments that in the end will produce a proof of concept and result in our final recommendations and will therefore conclude the thesis work.

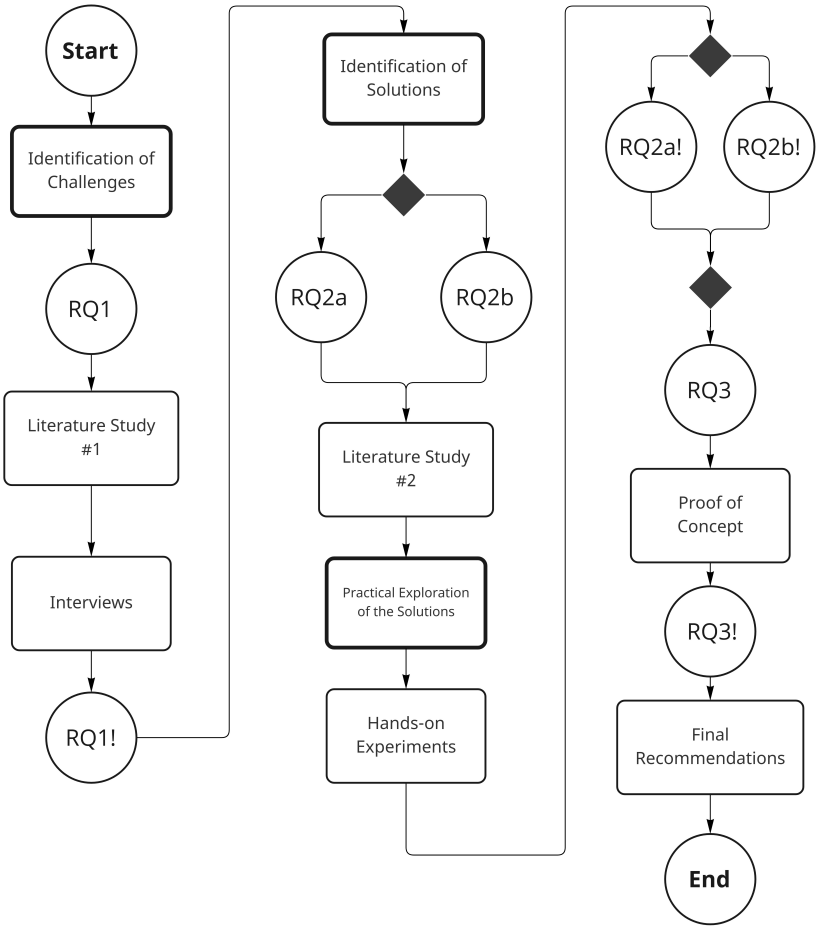

Figure 2.1: A flowchart explaining the planned methodology to be used. The rectangles with bolded borders indicate that a new phase is starting, the circles indicate that a specific research question is being investigated and is then concluded with an exclamation mark.

2.2.1 Identification of Challenges

Since the ultimate goal of our thesis is to provide recommendations for handling multiple CSPs, the necessary initial step is to identify the associated challenges which is done in RQ1. This activity requires a significant amount of data which we plan to gather by two separate activities, a literature study and interviews. Our motivation for performing a literature study is to collect a wide range of previous experiences, such as the most common and recurring challenges, from related studies. We consider a literature study to be the most appropriate and efficient method for identifying the challenges given the time constraint for the thesis. An alternative approach would have been to conduct an extensive questionnaire aiming to discover common challenges from various sources of information, such as experts of the subjects or corporations working with multiple CSPs. However, we consider this approach to be too uncertain and time consuming regarding finding suitable candidates and receiving their contribution in time even though the results could potentially have been more qualitative. Since the aim of this part is to gather quantitative data, a literature study is therefore considered the best option.

We also recognize a need for qualitative data to complement the quantitative data gathered by the literature study. We plan to address this need by conducting interviews with developers at the IKEA team. The reasoning for performing interviews is multi-folded; the first reason is to reveal potential practical challenges that have not been addressed by the literature. The second reason is to get a more elaborated reasoning and discussion regarding how the common challenges from the literature study present themselves in the practical environment at the IKEA team. The final reason is to collect data from another source in order to be able to compare and validate the outcome of the literature study. Alternative approaches could possibly be to conduct a questionnaire, case studies or interviews outside the IKEA organization. A questionnaire is however not considered sufficient enough, as this data gathering method does not provide an opportunity for discussions and potential follow-up questions which we consider a necessity for the qualitative data gathering. Our motivation for not gathering qualitative data outside the IKEA organization is that given the scope of the thesis this would be too time consuming and would require us to reprioritize the entire thesis work. Also, due to the size of IKEA, there exists a wide range of different teams and experiences within IKEA, all in different phases of maturity regarding cloud development that could assist us in our work if needed. Therefore the vast spread of both knowledge and experiences within the IKEA organization is considered sufficient for our intended purpose.

Literature Study 1

In order for the literature study to be conducted in an effective and efficient manner, an approach for finding relevant literature needs to be established. Our plan is to use a combination of both individual and collective research to utilize our collective competence as efficiently as possible.

The first part of the literature study is to brainstorm potential keywords and search terms to use. Once an initial set of search terms have been created it will be used to individually search for literature by using Google Scholar and LubSearch (a collective entrypoint to all of

Lund University libraries' resources). These search engines are considered reliable platforms for providing trustworthy and quality literature references for the thesis. We then plan to create two separate reference lists that we will individually compile. The two lists will consist of literature whose title and abstract are recognized as the most relevant and interesting ones for the given topic. The two lists are then to be compared to each other and references that are present in both lists will be prioritized, but all references will be discussed and evaluated based on their abstract. The idea is that references present in both lists are deemed to have a higher probability of being interesting and relevant for the thesis and therefore should be evaluated prior to the others. Other aspects that are taken into account when prioritizing the literature is the year of publication and how many other studies have cited the paper. Due to fast development and the vast amount of research on the subject some literature can potentially be considered outdated. Therefore, the prioritization criteria for relevant papers will be adjusted throughout the literature search according to our findings and needs. The papers that pass the filtration criterias will then be read and important parts and keywords will be summarized in external documents. The purpose of these documents is to gather the important information from each paper and prevent a loss of information over time. Each document will then be categorized according to the challenge the paper is describing and this will result in a small encyclopedia containing all the challenges related to handling multiple CSPs and corresponding work describing it and thus contributing to answering RQ1.

Whilst reading the literature, new search terms and keywords are most likely to emerge and these will be noted and used in future literature searches. Another approach for finding new search terms and keywords will be to simply search for current keywords on Google and find non-academic articles such as blog-posts and popular science articles. Although some of these findings can be deemed unreliable due to a lack of references and trustworthiness, they can still provide new angles of approaches that can be used to find additional academic literature in the search engines described above. A final approach for finding new related literature will be using the *snowballing approach*, both forward and backward, on current literature found [6]. Snowballing backwards means that the reference list of a paper is investigated for other sources of information that could contribute to the thesis. Snowballing forward is referring to the use of LubSearch and Google Scholar's "cited by" function to find newer papers that have used the current one as a reference.

All these steps are then planned to be performed in an iterative manner until the amount of new findings begin to stagnate and a sufficient amount of data has been collected. The initial set of search terms used for finding literature is presented in Appendix A.1.

Interviews

For our given purpose of finding qualitative data by conducting interviews, we consider semi-structured interviews to be the most appropriate approach since they provide space for discussions and a possibility for discovering other angles of approaches [7]. They also ensure that the relevant topics are covered by the predetermined questions, an approach we consider will suit this stage of the data gathering process well. Therefore the initial step is to formulate interview questions that ensure that all topics we want to discuss with the developers from the IKEA team are covered. An initial set of get-to-know-the-person-questions whose purpose

is to establish a relationship with the interviewee as well as finding out their background and knowledge of the subject are going to start off the interview. The detailed, more open-ended and topic-related questions are then planned to follow. Follow-up questions will also be prepared in case the interviewee's answers are considered to be non-adequate or too brief. Due to the nature of semi-structured interviews, other spontaneous questions can also be asked depending on the direction of the interview [7]. All interviews will be recorded, given that the interviewee gives the permission to do so. The recording will allow us to fully focus on the interview and the conversation with the interviewee instead of being forced to take notes and potentially disturbing the flow of the conversation. Also, this will provide us with the opportunity to go back and analyze the answers in later stages of the thesis.

In order to find suitable candidates for the interviews, the first interview is going to be conducted with a cloud-expert, who also happens to be the IKEA supervisor of this thesis. The intention is that this will result in both feedback on how the questions could be further refined as well as suggestions of appropriate people with varying roles and backgrounds to interview next within the IKEA team. These candidates will then be approached and interviews with each one of them will be scheduled. Since the interviews are going to be held at various stages between week 9 and 11 of the thesis, the opportunity to refine the questions between sessions are available. The final version of the interview questions is found in Appendix B.

Concluding the Identification of Challenges Phase

Once the literature study and the interviews have been completed, we should have collected a sufficient amount of data regarding RQ1, the challenges associated with handling multiple CSPs. The data from the literature study needs to be analyzed and compiled before categorizing the challenges according to their background, nature and relevance. The same procedure is then planned to be executed on the findings from the interviews as well. When the two different sources of challenges have been processed, we plan to compare the results and reveal similarities and differences between them. The expected result is to both find common patterns between the literature and the context of the IKEA team as well as to determine which challenges are most relevant to further explore for them. This will conclude RQ1 and serve as the foundation for the remaining research questions where the solutions to the identified challenges will be investigated.

2.2.2 Identification of Solutions

Once the challenges of handling multiple CSPs have been identified and RQ1 has been concluded, the search for suitable practices and alternative solutions will begin. This phase of the thesis will aim to answer RQ2a and RQ2b. Accordingly, we plan to start off by identifying potential solutions for the problems revealed by RQ1, as stated in RQ2a. In parallel we will explore RQ2b, whether or not a variant perspective can result in a suitable solution for handling the challenges identified in RQ1. Once again, we consider a literature study to be a suitable choice for our intended purpose as we want to find multiple potential solutions discussed in previous work. This choice will be further motivated in the final subsection of this section.

Literature Study 2

The purpose of this second literature study is to identify potential solutions and practices for handling multiple CSPs described in literature. We considered the articles found in the initial literature study to be a suitable starting point for investigating RQ2a. A majority of these articles did not only present problems related to handling multiple CSPs, but also introduced some kind of potential solution for their presented problem. By utilizing the findings from literature study 1 as a starting point, the time spent searching for literature will be greatly reduced in literature study 2 while still covering a broad range of potential solutions. However, the remaining parts of this literature study, including finding new search terms and keywords, will follow the same approach as literature study 1 discussed in section 2.2.1.

In order to explore RQ2b we plan to use a different approach to find an initial set of articles and keywords. We will use the literature list from the Configuration management course to identify sources of information related to the variant management perspective. From these articles, we will be able to continue with the snowballing approach for finding additional literature as discussed in section 2.2.1. The initial set of search terms used for this literature study is presented in Appendix A.2.

2.2.3 Practical Exploration of the Solutions

The final phase of the thesis aims to explore the results from the previous phases in a practical way. This will be achieved by conducting experiments on applications that resemble the IKEA team's context. Our expectation is that these experiments will help us discover advantages, disadvantages and possible difficulties when adopting the solutions found. In other words, once the experiments are concluded, we will have gained experience regarding which challenges should be given extra attention and which solutions seem to be the most suitable ones for the IKEA team and we will therefore be able to conclude RQ2a and RQ2b. The experiments will also produce a proof of concept which will conclude our final research question, RQ3. Further motivations for conducting these experiments will be discussed in the final subsection of this section.

Hands-on Experiments

The purpose of the hands-on experiments is to gain experience in cloud development in order to explore the severity of the challenges identified as well as the practical suitability, advantages and disadvantages of the solutions discovered in the previous literature studies. In order to do so, we plan to first get more experience working with CSPs and GCP in particular. Our IKEA supervisor advised us that a good starting point would be to complete the *pic-a-daily serverless workshop*, as it touches upon a wide range of the cloud resources used by the IKEA team [8]. The pic-a-daily serverless workshop is a lab series offered by Google where an application is developed by using several different GCP services. The final application allows the user to upload a picture on a GCP hosted website. The picture is analyzed by a Google API and the characteristics of the picture is displayed together with a collage of all the latest uploaded pictures before the results are stored in a Google database. The user is introduced

to several GCP resources such as triggers, buckets, service accounts and messaging services among others. We plan to follow our supervisor's recommendation and complete the lab series which should result in both experience as well as a small project that we can use in our research and experiments.

The next step is to evaluate the solutions identified in the second literature study by conducting an experiment. The experiment will consist of writing a small application that will be based on the previous experience from the pic-a-daily serverless workshop but will be scaled down to fit the scope of the thesis and only include cloud specific parts that are relevant for the IKEA team. Our motivation for conducting this experiment is to both evaluate theoretical solutions as well as finding the best practice for handling multiple CSPs. In order to achieve this, several objectives have been defined. The first objective is to get the application up and running on GCP. In this stage, we assume the application will be heavily vendor-locked to GCP. The next objective is get the application working on another CSP. Due to the IKEA team's interest in Alicloud, we have chosen this CSP for our experiment. This will require us to make necessary changes in the code to escape the GCP vendor lock and as a result enable a smooth porting of the application to Alicloud. At this point we expect to have two different variants of the same application meaning that we will have to use our findings from literature study 2 to handle and maintain these variations in an efficient and practical way. The final objective is to evaluate if our modifications have reduced the vendor lock-in so that support for additional CSPs can be implemented without any further complications. We plan to do this by adding support for Microsoft Azure (Azure) and comparing the difficulties and implementation time to the Alicloud implementation. This is important to explore as some solutions may require an extensive amount of work to initially implement but will allow additional CSPs to be added with no additional work, while other solutions may require a similar amount of work for each CSP added.

2.2.4 Motivation for Our Chosen Methodology

We have identified alternative approaches for the solution identification and evaluation processes but for several reasons we have chosen the previously discussed combination of a literature study in conjunction with hands-on experimenting to be the most suitable approach. One alternative approach that was considered was to conduct a literature study for both the solution identification and the evaluation of solutions processes. Our reasoning for not choosing this method is that it would be difficult to find relevant and reliable research for the context of the IKEA team. Most likely, a larger number of possible solutions would have been revealed but the evaluation of them in the context of the IKEA team would be missing which would leave no guarantee that the solutions were suitable in practice or that all advantages and disadvantages would be revealed. This uncertainty on the reliability of the evaluation would most likely result in the final recommendations provided to the IKEA team to be less useful. On the opposite end, a different approach would be to not conduct a literature study and simply start experimenting on the code provided by the IKEA team to reveal potential solutions. However, even if we were considered experts on the subject, this approach would produce a biased result and most likely result in potential solutions being left unexplored. A final alternative for identifying and evaluating possible solutions could be by conducting interviews, case studies or other similar activities at other organizations that have faced similar

challenges as those found in RQ1. Even though this would most likely produce a trustworthy result, this approach is deemed to not fit the scope of the thesis. Finding suitable candidates in itself would require a significant amount of time and effort as these kinds of challenges are often not visible for external parties outside the organization. Assuming we would find appropriate candidates and that they agree to be part of the thesis, it would still be difficult to produce results of high quality within the limited time frame for the thesis. With all the aforementioned alternatives in mind, our methodology of choice is a combination between a literature study and practical exploration through hands-on experiments. We believe that the literature study is an important part for identifying a wide range of potential solutions whilst the hands-on experiments are important for a proper evaluation of the solutions practicality in the context of the IKEA team. We deem this approach to have the highest chance of producing recommendations of quality to the IKEA team.

2.3 Theory

The purpose of this section is to expose the theoretical foundation and introduce the reader who is unfamiliar with cloud computing to necessary background theory that we consider vital in order to comprehend the remaining parts of the thesis. For the experienced cloud user, this section is also of interest since it will clearly define important concepts that are unambiguously used throughout literature. The thesis will address issues related to variant handling and hence some fundamental variant theory is also introduced.

2.3.1 Cloud Computing

Cloud computing is an agile model where organizations can get on demand access to a shared pool of configurable computing resources. These resources can rapidly be provisioned according to the needs of the customer and once they are no longer needed they can be released with minimal effort [9]. This multi tenancy model enables greater utilization of the resources for the CSP which enables them to offer these resources to a reduced price for the customers. The ability to get on demand resources also enables customers to better handle peaks in their capacities without the need to invest in on-premise equipment that meets peak demands [10]. Today there are several different CSPs available, where the most widely used are Amazon Web Service, Azure, GCP and Alicloud which together account for 63% of all cloud spendings world-wide [11].

Cloud computing can also be seen as a form of outsourcing where companies use external CSPs to get access to a multitude of cloud resources such as computing power and storage [12]. This means organizations can focus on their core business and no longer need to maintain and manage on-premise servers in order to host their applications. However, cloud computing is a relatively new type of outsourcing where each CSP is offering their own range of proprietary services which have been developed with no uniform standard in mind [12].

2.3.2 Interoperability and Portability

Interoperability and portability are two commonly discussed concepts in the context of cloud computing and multi-cloud computing. However, since these terms are closely related to each other they are often used interchangeably and not always used correctly. It is therefore important to define these concepts separately to avoid misunderstandings. In the literature it is a common theme to use interoperability collectively for all terms related to both interoperability and portability. Hence, a clear separation between those concepts is required. Throughout the rest of the thesis, the concepts will be used with the following definitions in mind.

We have chosen to use the definitions presented in the article by Kolb and Wirtz which defines interoperability as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" [13]. The same article also defines portability as "the capability of a program to be executed on various types of data processing systems without converting the program to a different language and with little or no modification". In short, interoperability facilitates communication between multiple CSPs, while portability enables migration between different CSPs without any major adjustments to the application.

2.3.3 Cloud Native and Cloud Agnostic

Two important concepts that are used in the thesis that we consider necessary to introduce are *cloud native* and *cloud agnostic*. As illustrated in figure 2.2 below, cloud native can refer to both a design and an implementation context whilst cloud agnostic is referring solely to an implementation context [14].

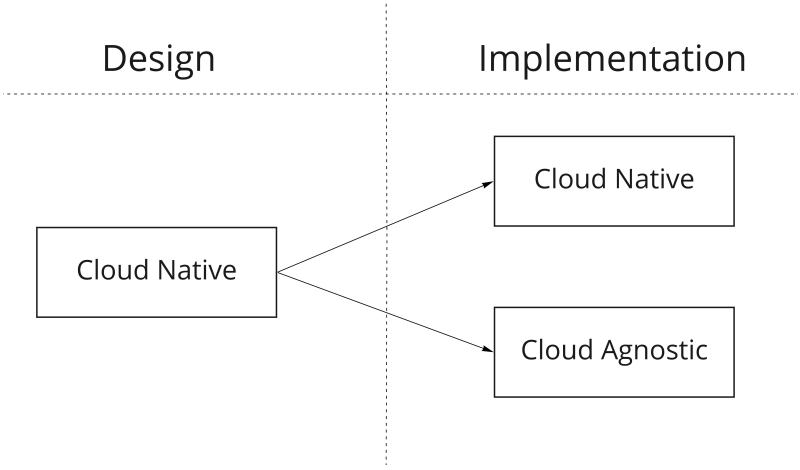

Figure 2.2: Visualization of how the concepts cloud native and cloud agnostic relate to each other.

A cloud native design refers to that architecture and design aspects of software aim to create new software that take the characteristics of cloud computing into consideration to utilize benefits such as elastic scaling [15]. In other words, the application is designed to be run in a cloud environment but there is no commitment to a certain CSP or even between public or

private clouds.

However, in the implementation phase a cloud native and cloud agnostic implementation can be seen as opposites of each other. A cloud native implementation is dependent on the services of a specific CSP and fully utilizes them. On the other hand, a cloud agnostic implementation is the implementation of a cloud native design that is not dependent on the underlying CSP services, except for the runtime [16]. This means abstractions are used to only utilize the least common denominator and unique CSP services can not be used.

2.3.4 Variant Handling

Although we do not yet have the details of how to handle multiple CSPs, we do know that the same application hosted on multiple CSPs will result in variants of the same application to some extent. The application part of the code will be similar across all CSPs, while other parts are very specific to the CSP the application is hosted on.

An important aspect to have in mind when discussing different kinds of variants is the *double maintenance problem* first introduced by Wayne Babich back in 1986 [17]. The double maintenance problem arises when there are two or more copies of the same software, meaning that all copies need to be kept up to date with each other. Babich states that eventually someone will forget to update all copies accordingly and hence, "multiple copies inevitably diverge". This may imply that double maintenance problem might become relevant when discussing potential solutions later on.

Axel Mahler does also acknowledge this problem but he refers to it as *the multiple maintenance trap* [18]. He writes that when adjustments are planned to be applied to those parts of a system that are common for several variants, but are for some reason separated, the multiple maintenance trap arises. When the number of variants and the frequency of changes between them increase, the variants will eventually get out of sync. A scenario similar to the one Babich is describing. Mahler does however address this problem by introducing two different variant handling concepts, variant segregation and *single source variants*.

Variant segregation can be described as maintaining a separate copy for each variant of a specific component. Mahler recommends variant segregation when there is only one dimension of variation and when the variant siblings have no or very few lines of code in common [18]. This means that components that are common across all variants only exist in one file while the components that differ are extracted into variant siblings and contained as their own file. This is not always easy to achieve since it can be hard to determine which parts are common between all variants and changes can also result in common parts being transformed into variants. There is also a risk of introducing the double maintenance problem if the variants include parts that are common between all siblings.

Single source variants is a method where all variants are contained in a single source object and the variants can be extracted when needed [18]. This is illustrated in figure 2.3 below, where the three variants of C are extracted from the common source object. The advantage of this approach is that the redundancy between variants can be almost completely avoided.

The major disadvantage is that the code can be difficult to read and comprehend as multiple variants are present in the same file which complicates the maintenance of the file. The method is however recommended when the amount of code that is common between the variants is large in contrast to the variant specific parts.

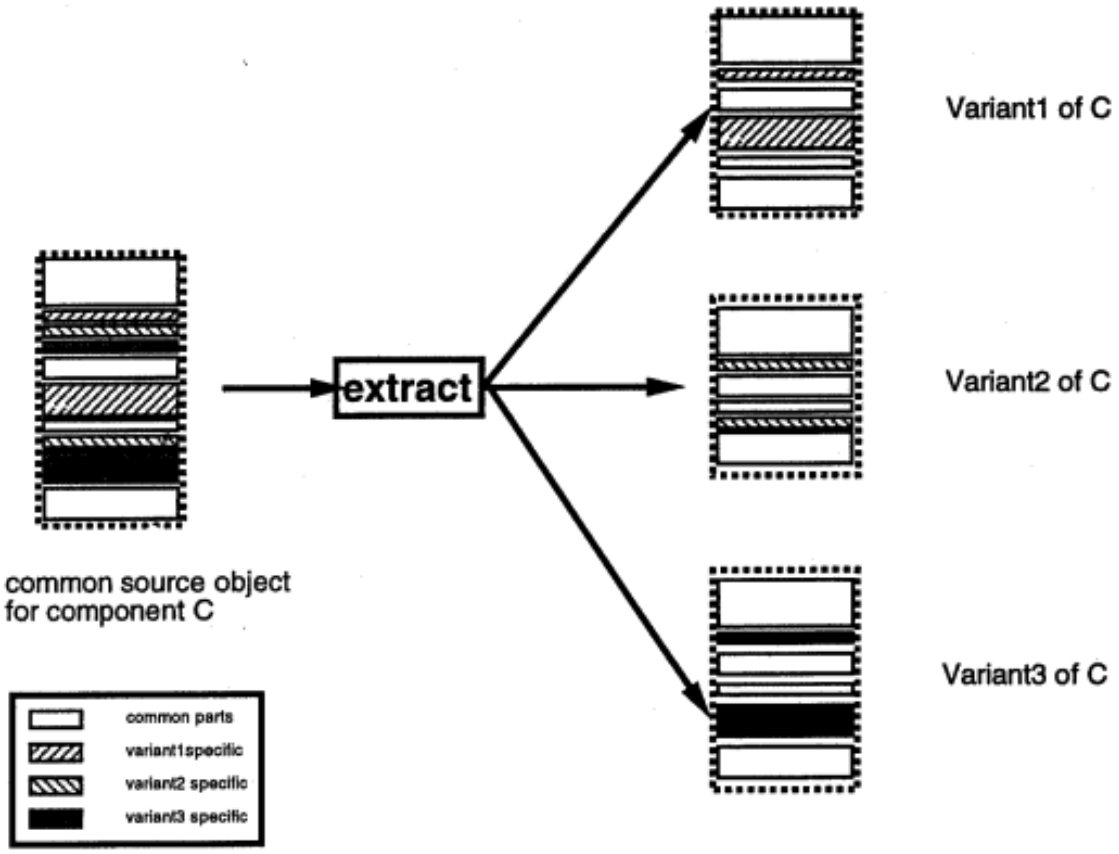

Figure 2.3: Single source variants illustrated by Mahler [18].

Chapter 3
Identification of Challenges

The objective of this chapter is to investigate RQ1 by identifying the most common challenges encountered when handling multiple CSPs. There exists previous research and knowledge about how to handle multiple CSPs which we aim to examine and discuss. This is done in order to reveal the challenges that the IKEA team will eventually need to handle when adopting a multi-cloud strategy. By first examining the findings from previous studies related to the topic of multi-cloud, previous experience is taken into consideration. Then the IKEA perspective and experience is addressed by interviews with employees at the IKEA team. Finally, in the last section of this chapter the findings from the two approaches are compiled and discussed in order to conclude RQ1.

3.1 Literature Study 1

The purpose of the literature study is to explore and investigate findings from previous studies related to the challenges associated with handling multiple CSPs. Our expected results are to reveal the most common challenges and also discuss a few different angles of approaches that have been argued in the literature. We have read and summarized 75 articles in total and an additional 40 articles were briefly read but not summarized or referenced due to their scope or lack of new information. All these articles contributed actively or passively with references and knowledge about the subject which we have used to write the following sections. All the literature referenced in the literature study was found using the methodology presented in section 2.2.

Once the literature study had been concluded we realized that we had a vast spread of challenges and we considered some of them to be out of the scope of the thesis. Therefore we decided to categorize the identified challenges into two main categories, technical and non-technical challenges, where the technical challenges were the ones we deemed to fit the scope of the thesis. The non-technical challenges are still highly relevant when adopting a multi-

cloud strategy and will therefore be discussed in this chapter, but addressing these challenges further is outside the scope of the thesis.

3.1.1 Technical Challenges

The literature study revealed several technical challenges that we could group into four different subcategories. In the following subsection we will introduce each subcategory and discuss how different articles present and discuss the challenge. This will allow us to get a nuanced view of the current state of the problem which we need in order to investigate how they can be addressed later in the thesis.

The Lack of Portability

Portability is defined as the capability of a program to be executed on various types of data processing systems without converting the program to a different language and with little or no modification [13]. The following section will discuss why the lack of portability within cloud computing hinders companies from adopting a multi-cloud strategy.

As the world is in constant change, businesses need to adapt, and possibly reevaluate decisions that already have been made. This is essential in order to respond to the new prerequisites given and still remain a competitive alternative on the market [19]. Bozman and Chen states that once an organization decides to move between CSPs, it is essential for them to both know *how* to switch and what the *cost* of doing so would be [20][21]. Furthermore, given that cloud computing still is a relatively new phenomenon, there is a lack of guidelines available for how switching should be done. Petcu and Vasilakos describe portability as the largest obstacle hindering an increase of cloud adoption [22]. Opara-Martins et al. emphasize that portability is both desired and expected by the customers of the CSPs as they want to fully utilize business opportunities presented to them as well as avoiding outages [12]. Opara-Martins et al. further explain that cloud computing can be seen as a form of outsourcing, given that workloads and data are moved outside of the organization. Regarding outsourcing, the most important aspect to keep in mind for the customers is how to exit the contract if needed. Hence, portability should be a key criteria for any customers looking into cloud services. However, this is most often not the case due to the current market state of cloud computing. Although customers would prefer portability, there is a lack of standardization across the CSPs' offerings given that portability is in fact not in the CSPs' best interest [22][23][24][25][26]. Elkhatib states that the major CSPs see standardization as a solution to the portability problem that would give their customers an opportunity to effortlessly switch to a competitor, which is not something that benefits them [27]. Currently, the major CSPs prefer unique features that distinguish them from competitors rather than solely having commodities with no clear competitive advantage. For the cloud customers, this means a range of different features to choose from but it also results in a lack of portability when they have committed their business to a certain CSP.

Portability can be divided further into both data portability and application portability, each with their own specific challenges [28][29].

Data portability is the ability for customers to migrate data between different CSPs [28][30]. The common challenge here is once again a lack of standardization, which for example results in differently structured databases and that import and export functionality are working differently. There can be different syntax for the source and destination of the migration as well as semantically differences. If the semantics differ between the source and target, the ability to migrate data can be impossible in some cases [30][31]. Bozman explains that simply being able to move data between CSPs will not solve the data portability problem [20]. Most applications require data to be formatted in a certain way and CSPs will need to support conversion between data formats as well as support for compatible storage services in order for this issue to be solved. Gonidis et al. agree with Bozman in the way that simply migrating the data is not sufficient for achieving data portability since conflicts can occur at later stages such as when data querying is done in different languages or incompatible data structures are used [31].

The application portability problem refers to the issue that often arises when an organization decides to either switch CSP or integrate a solution from another CSP into their business [30]. More specifically, applications are not possible to migrate since the semantics and services between the CSPs differ and can not be used across competitors' alternative solutions. Certain comparable services from each CSP even support different programming languages which can make a migration impossible [31]. The semantic differences in naming terminologies between CSPs also creates portability challenges for the cloud customers [32]. The challenges are present when the customer mistakenly assumes that a service that shares the same name also shares the same functionality. Bozman and Chen state that CSPs are eager to provide tools and APIs to ease the migration onto their platforms and import standardized data into their proprietary solutions [20]. However, from a customer perspective the process of moving away from adopted solutions or between CSPs is a much more complicated process. The reason for this is that the tools and the APIs provided by the CSP are generally proprietary and specific for their platform which obstruct the process of migration to another providers platform [31][33][34][35][36]. In most cases, migration between CSPs therefore generally requires a complete reimplementation of CSP specific parts of the application. Yasrab and Gu state that incompatibility of APIs makes it impossible for customers to switch between different CSPs [37]. In general, cloud applications are developed using several APIs and libraries that are specific for a certain host environment [38]. These resources are often hardcoded and invoked within the source code of the application itself, complicating the migration between CSPs.

The Lack of Interoperability

Interoperability is defined as the ability of two or more systems or components to exchange information and to use the information that has been exchanged [13]. This section discusses why the lack of interoperability becomes a problem for organizations adopting a multi-cloud strategy.

A study done by Oracle concluded that organizations abandoned on average one cloud application each year because integration problems got out of hand [19]. The study further revealed that 81% of the participating organizations agreed that it is essential to have cloud

applications entirely integrated with each other as well as other systems in the organization. This result is further supported by the article of Dillon et al. where the authors define that the scope of interoperability includes both the connection between the CSP and the local on-premise system but also the connection between different CSPs [39]. They further state that the main goal of interoperability is to enable a smooth exchange of data between cloud applications. This would increase resilience of the cloud system as a whole and would be a way to increase competition in a market dominated by a few CSPs [12]. Customers of CSPs also benefit directly from interoperability since they are able to compare and choose between offers across different CSPs, resulting in the possibility of "best of breed" solutions for all applications [12][25].

Similarly to the portability aspect, the lack of standardization is a present challenge regarding interoperability as well [40]. Several authors describe that the main difficulty for achieving interoperability between different CSPs is the lack of agreed upon standards for protocols, interfaces and data formats [12][41][42]. Ferry et al. elaborate upon the reasoning by implying that the lack of standardization decreases the interoperability between the services of CSPs and hence, ultimately hindering the optimization of cloud efficiency [43]. Opara-Martins et al. point out that the number of different stakeholders involved in this area, with different views and opinions, can potentially harm the development of a unified standard [30]. This could lead to several standards being developed which in turn amplifies the originating problem even further.

In the article written by Ranjan, he explains that heterogeneous APIs are a major problem regarding the aspect of multi-cloud interoperability [32]. The present solutions from the CSPs are usually not compatible with each other [39]. Rather, they tend to have a proprietary API designed for their own solution which does not take multi-cloud functionality into consideration. The reasoning behind this is that having their own API simplifies their development by not having to adapt their implementations to anyone else [44]. Instead they can fully focus on maximizing the development of their own cloud solution and provide a solution as good as possible for their customers. Having different APIs across the cloud solutions also makes the utilization of the systems more difficult [45]. The same operations have to be developed and engineered for each CSP to ensure working communication channels, both with other cloud solutions and the organizations own existing systems [36].

Another interoperability challenge mentioned in articles by both Petcu and Vasilakos as well as Opara-Martins et al. is data synchronization [22][30]. When the same data dependent application is running on multiple CSPs, the ability to synchronize data between the clouds can be a vital requirement. This synchronization requires interoperability to exist between the two CSPs.

Vendor Lock-in

The vendor lock-in problem arises when the severity of the portability and interoperability challenges get out of control and the switching cost to another CSP is deemed too high. The customer is then locked to their current CSP with no reasonable opportunity to restructure their current solution and use an alternative CSP. However, it is also possible for vendor

lock-in to arise from other aspects such as developer preferences and limited CSP knowledge [46].

In 2013, the European Network and Information Security Agency and European Commission recognized vendor lock-in as one of the greatest organizational challenges for cloud adoption [30]. Hong et al. explain that the vendor lock-in issue is closely related to both the portability and the interoperability challenges [47]. Opara-Martins et al. describe three different scenarios for how customers get locked into their CSP's solutions [12]. One aspect is when a CSP is deliberately designing their system to be incompatible with competitors. This is a case of CSPs utilizing the lack of interoperability between clouds to force customers to not only use one of their offerings, but all of them. The second aspect is when CSPs use proprietary standards that hinders customers from moving their applications to other clouds. By introducing proprietary APIs and libraries to the customer, the portability of the application will be greatly reduced, practically locking the customer with high switching costs such as rewriting the whole application. The third aspect is when software is licensed under exclusive conditions. This can legally lock the customer to the provider and not allow the customers to use any competitive solutions. Opara-Martins et al. explain that these aspects can potentially damage the growth of the entire cloud ecosystem by limiting organizations choice of CSPs, thus preventing a competitive and healthy market situation [12].

A study made in the UK by Opara-Martins et al. also identified security and vendor lock-in to a single CSP as major threats for cloud adoption [30]. The study also revealed that the reason for why vendor lock-in is considered such a challenge is that it creates a barrier for the customers for adopting cloud solutions in general. A multi-cloud strategy is generally not the initial step when first adopting cloud computing. However, the problem is that the alternative is to adopt a single CSP strategy which usually results in vendor lock-in that could become a problem in the future [48]. Another takeaway from the study was that less than half of the respondents stated that they had a fundamental understanding about the vendor lock-in phenomenon. This result validates the results from an article made by Lipton, where he explained that the complexity and cost associated with changing to a different provider is most often underestimated [49].

Another important aspect to consider when discussing the lock-in challenge is that there are different kinds of lock-ins, and getting rid of one often results in another type of lock-in [46]. For example using a third party solution to avoid being locked to a CSP, does in fact result in a new lock-in to the third party. Although this might not be a vendor lock-in, it is still at least a product lock-in. The question then is what type of lock-in the customers are willing to accept and how much they are willing to sacrifice in terms of efficiency and cost to avoid a specific lock-in? There are also other types of lock-in that could pose a threat for a business. Hohpe explains that even if a business manages to get past the vendor lock-in challenge, employees' skills might be locked to the former provider [46]. This will result in an additional cost for the organization in terms of the developers having to learn the new platform or alternatively even the need to hire new competence. Hohpe also states that there is a considerable risk that the developers are themselves locked to a certain provider, a mental lock-in, which is affecting their daily work by for example making assumptions based on their knowledge from working with the previous CSP.

Security Aspects

Several studies indicate that security is one of the major challenges of multi-cloud adoption [34][35][40]. Generally, deploying applications on multiple CSPs is a more complex and complicated task since unique parts for each CSP need to be implemented while not affecting the functionality of the application on other CSPs. This results in larger applications which in the end means that attackers have a larger attack surface and more possibilities to inflict damage which increases the possibility of security breaches of the organizational data. In a single cloud context, customers have the possibility to use expertise and tools provided by the CSP for dealing with security issues such as access management and data security. However, in a multi-cloud context the provider's tools do not necessarily work across different CSPs [50]. APIs between CSPs can differ and further complicate the deployment of the application on different CSPs [51]. Pearson and Benameur explain that since cloud APIs are not yet standardized, clarifying the roles of responsibilities between the provider and the customer is a challenge in itself [40]. If customers are unable to fully utilize the features offered by the provider, the security instead becomes the customers responsibility. Consequently, customers generally have to use third-party tools to manage security for applications deployed on multiple CSPs. Similarly the management of identity and access control as well as the secret management of encryption keys suffer from the same problems where the solutions provided by CSPs generally can't be used in a multi-cloud context.

Another aspect closely related to security is trust. Pearson and Benameur describe that a significant issue related to cloud computing is for the CSP to ensure that the customer has control of the entire lifecycle of their data, especially the deletion of it [40]. Organizations want to be sure that they are in control of their data and that there is no possibility for the CSP to recover and use the data after deletion. Pearson and Benameur mean that this issue currently relies on trusting the CSP. In a multi-cloud context this issue is amplified due to the large number of copies and involvement of multiple parties. The copies themselves are also a potential threat as providers might keep backups, sometimes even without the organization's consent, which increases the vulnerability from an insider or external attacker [40].

3.1.2 Non-technical Challenges

The literature study also revealed some challenges that are not directly related to the technical aspects of multi-cloud. These challenges are very important to consider when making decisions regarding a multi-cloud adoption but might not affect the individual developers to the same degree as the technical ones. Due to the scope of the thesis, these challenges will be introduced and discussed in the following sections but will not be further investigated in the remaining thesis as we have decided to limit our scope to only explore possible solutions for the technical challenges.

Legal Aspects

As for many types of legislation, the laws and the legal aspects for cloud computing and user data are closely related to the physical location. A multi-cloud strategy often results in utilizing data centers at different locations, in various countries across the entire globe. Different

laws may be applied depending on which country the data is stored in and more often than not, each CSP stores the data in multiple locations at the same time [40]. Handling this can already be considered a challenge when handling a single CSP and a multi-cloud strategy only reinforces the need of a structured plan on how to manage these regulations.

According to Hong et al. organizations tend to underestimate the challenges of obeying local laws in the system design phase of a cloud computing system [47]. Having a multi-cloud solution further complicates the situation since there are multiple physical locations and correspondingly multiple laws to consider. Ideally the challenge should be addressed early in the application's lifecycle. This means that the challenge could be even more severe when existing projects adopt a multi-cloud approach. The authors also mention that there are no uniform standards in place between CSPs for service level agreements, which can complicate the problem even further since customers can be held accountable for different areas depending on which CSP they are working with. Hong et al. predict that this will be one of the main challenges of multi-cloud computing in the near future.

Another aspect that has to be considered when having a multi-cloud solution spanning over several different countries is the privacy aspect of the data stored in the cloud. In some countries, governments have legal rights to access and view the data that is being stored in their physical jurisdiction [40]. Meaning that under certain circumstances, the government is able to access the data without the need to notify the cloud customer. Similarly, in some countries a CSP can be forced to hand over their stored data due to litigations or jurisdictional matters. Depending on the privacy level and sensitivity of the data, this is something CSP customers need to be aware of when expanding their businesses to CSPs located in new countries or regions.

This challenge can also occur the other way around. Instead of risking a third-party accessing and compromising the company data, organizations might not even be able to legally move the data outside the country of origin [46]. This means that organizations are locked to CSPs within the same market, and depending on the aim of the multi-cloud strategy, this might jeopardize the whole plan of expanding to new markets. Similarly, customers might not be allowed to move some systems to the cloud at all due to the terms of their software providers' licenses.

Sustainability

Sustainability is one of the century's greatest challenges and cloud computing is responsible for a big part of the world's energy consumption. According to Buyya et al. the total energy consumption of cloud computing in 2018 was greater than most of the world's countries, only surpassed by the four largest economies of the world [21]. As an illustrative example, a single data center requires on average the same amount of energy as 5,000-25,000 households and as of 2021 there existed more than 7.2 million data centers around the world [21][52][53].

Since sustainability is one of the most important questions in today's society it is important for companies to fulfill the sustainability expectations of all their stakeholders. A cloud strategy, especially one involving multiple CSPs, moves a big part of the energy consumption

out of the organization and into the cloud. Hence, the sustainability of the cloud customers is coupled to the sustainability of the CSP. It is therefore important, when adopting new CSPs, to consider how the provider's sustainability policy aligns with their own ambitions.

Economy

One of the main economic challenges when handling multiple CSPs is the ability to compare different providers' pricing models. As of today, the CSPs offer different pricing models and comparing offers between different providers is both time consuming and challenging since CSPs generally do not use the same terminology for corresponding services. If their services use the same terminology, it is not guaranteed that they are semantically the same product [21]. In other words, even though two services might share the same name, the details regarding throughput, pricing etc. may vary a lot between services.

Another difference that complicates the comparison between the offerings from the CSPs is the differences in the charging models. For a simple Function as a Service (FaaS), one CSP might charge customers based on the number of function calls, while another provider might charge the customers for the cumulated execution time. This makes a comparison rather difficult and will require a more detailed understanding regarding which CSP's offering will suit the application best from an economical perspective.

Talent Management

Binz et al. mention in their article that using external solutions from CSPs often result in a somewhat unexpected challenge for organizations; the lack of internal talent and knowledge [33]. They further explain that these external solutions pose a threat in terms of which external knowledge often must be acquired in order for them to fully understand and utilize the entire potential of the CSP's solution. Currently, advanced cloud skills are in high demand, and organizations are often competing against the CSPs themselves when it comes to hiring cloud developers. Therefore finding people with skills across multiple clouds is next to impossible, which is a clear issue when it comes to realizing a multi-cloud strategy [54].

3.1.3 Summary and Key Takeaways

In order to ensure that the reader understands which part of each challenge we consider most important, the purpose of this section is to summarize and highlight key takeaways of each challenge previously discussed.

Regarding the technical challenges, an important note is that portability is not in the best interest of the CSPs and they generally do not gain anything from making customers' applications compatible with competitors' alternative solutions. This means that developers are responsible for increasing the portability of their applications themselves. The main interoperability challenge can also be explained by the lack of standardized frameworks between the CSPs. For both the interoperability and the portability problem, the number of different stakeholders in the industry is harming the development of agreed upon common standards for a multi-cloud context.

The challenge of vendor lock-in is closely related to the interoperability and portability challenges but there are also other aspects such as developer knowledge and preferences that could lock organizations to a specific CSP. In many cases, resolving one vendor lock often introduces another type of vendor lock.

With respect to the security aspects of a multi-cloud development setting, the main challenge is that a single provider's security frameworks and tools do not necessarily work across different CSPs and generally, third-party solutions have to be used.

Regarding the takeaways from the non-technical aspects there are a few worth considering. For the legal aspects, laws and practices may differ between countries and CSP customers must be aware of this when adopting a multi-cloud solution for their organization. From a more environmental and sustainability based perspective of multi-cloud it is worth mentioning that cloud computing moves a large part of the environmental footprint to the CSP. From an economic perspective, the main challenge is to understand the differences in pricing and charging models and how they affect the total cost. Regarding talent management, the literature indicated that there is a lack of developers with multi-cloud competence.

3.2 Interviews

In order to better judge the severity and importance of the technical challenges we felt a need to gain an in-depth understanding of the situation of the IKEA team. We deemed interviews to be a suitable way of accomplishing this. At the same time, interviews would allow us to explore and expose the developers' experiences and concerns regarding adopting a multi-cloud strategy. We were also under the impression that interviews would complement the theoretical knowledge gained from the literature study with practical experiences. As mentioned in section 2.2, the plan was to interview developers within the IKEA team with different backgrounds and experiences from working with CSPs.

Our IKEA supervisor helped us identify suitable interviewees. In the end, five interviews were conducted where two candidates were DevOps-engineers and had a greater emphasis on the cloud infrastructure. The other interviews were held with software developers with various levels of cloud development experience and the interviews therefore mainly touched on the application level challenges of multi-cloud development. The previously conducted literature study influenced the direction of the interviews and as a result, the two main technical challenges interoperability and portability became the main topic of discussion. The next section will therefore discuss the IKEA team's perspective on these challenges while the following section will address other challenges identified during the interviews.

3.2.1 Main Challenges of Adopting a Multi-cloud Strategy

The literature study suggested that interoperability and portability aspects were the main challenges when adopting a multi-cloud strategy. This section will therefore discuss these

aspects from the IKEA team's current situation.

The Current Situation

The IKEA team's interest in a multi-cloud strategy is rather new, and has so far mainly focused on the infrastructure level. The interviews revealed that the majority of the developers have not yet adopted a multi-cloud strategy in development and are using GCP exclusively, meaning that the IKEA team is currently vendor locked. This is not strange since it is currently not a requirement to implement cloud agnostic solutions, although developers are encouraged by IKEA to investigate and consider implementing such solutions if possible. One interviewee mentioned that if there is a GCP solution available, very little effort is currently put into finding alternative agnostic solutions. There are however ongoing thoughts and discussions regarding further improving their applications by implementing cloud agnostic solutions.

The Portability Problem

The common perception between all the interviewees is that portability is the main problem for the IKEA team regarding their ambition to use multiple CSPs and become cloud agnostic. It became obvious that portability is regarded as a much bigger challenge to handle for the IKEA team than any other challenges mentioned; "Porting to another CSP would require us to thoroughly examine the majority of our current solutions. The application is currently not written in such a way that a swap would be effortless". The interviewee further clarified the reasoning by describing that the applications as of today can not simply just be deployed on another CSP. Instead, they would have to replicate parts of the application by using equivalent solutions offered by the other CSPs. This concern is shared among all the developers who state similar portability concerns. One interviewee even stated an almost identical concern, that the possible future challenge when deploying an application to another CSP would require a significant amount of research and resources, due to the fact that the applications of today are tightly connected to their current CSP, GCP.

Another aspect of portability is the deployment process. One DevOps engineer stated that deployment is the main challenge when it comes to portability. In the current setup, different kinds of cloud resources are used. The engineer explained that the main challenge is that they are not cloud agnostic, neither the tools, the applications nor the resources. The application is however cloud native and can probably be modified to work with different CSPs.

An effort made by the developers when it comes to prepare for multi-cloud is to containerize their applications by using Docker containers. One interviewee explained that by containerizing the applications, it is possible to run the container independently of the CSP. Before, they were using Google app-engine which resulted in a heavy vendor lock since the corresponding app-engine solutions offered differ between the CSPs and are in many aspects unique for each CSP. By containerizing the application and using GCP's cloud run, a more standardized service for hosting containerized applications is achieved. Similar services exist on the majority of the CSPs and the application is therefore not as tightly coupled to GCP anymore. This eases the deployment of the application to multiple CSPs but there are still challenges such as modifying pipelines for automatic deployment across multiple CSPs,

which is desired by the IKEA team.

One interviewee also mentioned that some of the applications are entirely dependent on the data and porting the data is a requirement for the application to be able to run on another CSP. A final takeaway that one of the interviewees also mentioned is that "it is not always worth the cost of being completely cloud agnostic. Sometimes a provider specific solution is better and worth the lock-in."

The Interoperability Problem

"Interoperability is not a big concern for IKEA at the moment" states one of the interviewees. The overall impressions when it comes to interoperability challenges at IKEA are that they are insignificant compared to the other challenges presented. In some cases, applications running on other clouds will only need data from that region and have no real need to communicate across CSPs. In other cases, the applications are already developed in a cloud native way, using REST-APIs to communicate. This means that communication between CSPs should not be a problem at the moment. One possible challenge that was revealed from the interviews is keeping data synchronized between CSPs if they target the same market. In the case of Alicloud being used for the Chinese market, all the relevant data would be there and synchronization between for example Chinese and European data would not be needed. But for example in the case of moving the Swedish market from GCP to Azure, the data would need to be synchronized between the clouds as some of the applications are very data dependent.

3.2.2 Other Challenges Identified

During the interviews a number of challenges not directly linked to either interoperability or portability were mentioned. A majority of the following challenges were touched on by several of the interviewees.

Security Aspects

One aspect regarding development in a multi-cloud environment that an interviewee pointed out to be especially difficult was security. With an agnostic approach in mind, the interviewee described that monitoring is often done with the CSP's proprietary solutions and making the monitoring solution 100% agnostic can be difficult to achieve. Therefore in a multi-cloud environment, coordinating each of the vendor-specific solutions becomes a challenge that has to be considered and dealt with. Similar concerns are present in other security aspects such as secret management.

Access Management

A concern raised by several interviewees was the number of accounts that a multi-cloud solution most often results in. An interviewee specifically mentioned a situation where employees did not initially get the correct access to a tool even though they were entitled to it. This

happened because the access management of the tool was separate to the overall access management and employees had to be given access in both systems. Although this issue should only arise once for every developer, it could still be considered a continuous problem that could become costly for large organizations such as IKEA, where new developers are hired regularly. This is a situation that best can be described by the double maintenance problem previously discussed in section 2.3.4, where the same action is required to be performed multiple times for each copy or, in this case, for each access management system. With multiple CSPs, there could be multiple access management systems that need to be handled in an efficient way to not hinder development. There are also service accounts for the different CSPs that need up-to-date tokens and keys in order to deploy the applications. These keys need to be managed and rotated in a way that is not hindering development.

Talent Acquisition

"It is very hard to find people with knowledge about the cloud". This reasoning by one of the employees reveals another challenge present for organizations like IKEA, acquiring talent. Almost all of the interviewees were under the impression that knowledge about cloud computing in general and specifically about working with multiple CSPs is hard to find. Currently, employees at the IKEA team are mainly using GCP and hence available training sessions and courses are mostly focusing on GCP-specific training and not so much on the alternative CSPs. When a multi-cloud strategy is fully adopted, developers would need training in the other CSPs' solutions as well in order to fully comprehend the differences. This is a switching cost that is often overlooked. Since most developers come in contact with the CSPs in one way or another, the total amount of hours of training can be significant for each new provider adopted. The common perception between the interviewees was that cloud development is complex and that lack of training can lead to costly mistakes.

Geopolitical Challenges

As previously mentioned, there are certain laws and regulations that force organizations to adopt certain practices and providers if they want to be present all over the world. Almost all of the interviewees discussed this issue with being forced to use specific CSPs for certain markets. An interviewee said "It gets very complicated when certain data is prohibited from being stored outside a specific country". In some cases, this could also include source code not being allowed to be executed outside the country. Hence, for organizations like IKEA that strive to be present globally, these concerns need to be addressed and are not always trivial to solve.

3.3 Comparison and Discussion of Challenges

There is a need to clarify the similarities and differences in the perception of the challenges severity between the literature and the context of the IKEA team. This will allow us to determine which challenges are most relevant for the IKEA team and therefore which challenges we should prioritize exploring solutions for in the next chapter. The purpose of this section is therefore to compare the findings identified in the literature study and in the interviews as

well as further discuss the results. First we will discuss the differences in the perceived severity of the interoperability and the portability challenges. We will also introduce an aspect we feel has been overlooked but should be highly relevant. Then, the other challenges are briefly discussed and compared in a similar manner. Finally, a motivation regarding which challenges we will prioritize in the upcoming chapters and our recommendations regarding the non-technical challenges will be given.

3.3.1 The Interoperability and Portability Challenges

Both the literature and the developers at the IKEA team seem to agree that portability is the main challenge when it comes to realizing a multi-cloud strategy. The developers at the IKEA team were under the impression that migrating their applications to another CSP would require an extensive amount of rework. This can be confirmed by the literature which highlights the lack of standardization in cloud computing in general. The literature suggests that this lack originates from the fact that it is not in the CSPs interest to introduce commonly agreed upon standards. This also means that the challenge will not be solved by the CSPs in the near future and therefore the challenge has to be handled by the IKEA team. Due to the amount of rework required we consider portability to be a highly expensive challenge to resolve.

The literature and the IKEA team do however disagree on the severity of the interoperability problem. The literature considers it to be one of the major challenges when adopting a multi-cloud approach. They motivate this claim by once again referring to the lack of standardization between different CSPs, which in many cases is hindering interoperability. In most multi-cloud adoptions, it is required that services hosted on different CSPs can communicate with each other. This is where interoperability is needed but the IKEA team is under the impression that their primary use of multi-cloud strategy would not require this communication. Because of this reasoning, they do not consider the interoperability challenge as their main priority when it comes to realizing their multi-cloud ambitions.

The literature does however include data synchronization as a form of interoperability challenge. The IKEA team may not need to synchronize data across different CSPs when they are used for different markets as customer data is only relevant for the specific region. However, when multiple CSPs are used for the same markets as backups or other reasons, there is a need to synchronize the data in order for it to be available on all CSPs. This could impose an interoperability challenge that the IKEA team is currently overlooking. We are under the impression that this challenge could become an expensive one due to the costs of constantly moving data between CSPs which is required in order to keep data up-to-date.

An Overlooked Aspect

With our current knowledge and experience from the university and more specifically the Configuration management course, we know that using multiple variants will always introduce the double maintenance problem to some extent. In the interviews, but mainly in the literature study, we felt that the challenges discussed were mainly considering the implementation phase. There were only minor concerns of the future maintenance of the multi-cloud

implementations. We believe that it is highly important to investigate how the future maintenance is impacted by the challenges and how it possibly could be facilitated. This is therefore something that we plan to explore and investigate in the following chapters.

3.3.2 Other Challenges

A non-technical challenge that was discussed both in the literature study and in the interviews was the problem related to the talent management in cloud computing. The literature argued that it was hard to find developers with experience working with multiple CSPs and that these people were in high demand and often employed by the CSPs themselves. The interviewees had a similar perception of the current state of the market, and agreed that developers with multi-cloud competence are generally hard to find. The developers at the IKEA team explained that their knowledge about working with cloud computing was in general limited to the knowledge gained from the courses about GCP at IKEA. This is a problem that we recognize ourselves, as there are almost no courses available that address cloud computing within our software engineering program at Lund University. This indicates that it is a common phenomenon that developers generally lack cloud computing knowledge, and especially multi-cloud competence, unless they have experience from previous work or by own initiatives.

Security aspect is a challenge that the IKEA team considered more severe than the literature. Both of them recognize security as a challenge for a multi-cloud adoption as CSP specific solutions are not functional between different CSPs. For the IKEA team the security aspects are vital for their entire business and introducing a multi-cloud strategy is only viable if the security aspects are not compromised. A multi-cloud adoption could impose a threat to the overview of the applications if for example the CSPs' monitoring solutions are used. This would require IKEA to adopt a third-party solution in order to retain the overview of their applications.

Another challenge that the IKEA team considers a bigger threat than the literature is access management. The interviewees described a need for a unified access management system in order to avoid a loss in productivity due to missing or differing access levels for different CSPs. This challenge is also something we experience ourselves when conducting this thesis. In a large company such as IKEA, getting access to certain services can take time and hinder progress.

3.3.3 Recommendations So Far and Challenges Addressed Going Forward

Due to the limited time and resources for the thesis, we will in the remaining chapter only focus on the technical challenges and their potential solutions. This does however not imply that the non-technical challenges can be disregarded when adopting a multi-cloud strategy. We will therefore give our final recommendations to the IKEA team for how these non-technical challenges should be further explored. This section will also include a discussion and motivation for how we will prioritize the exploration of the technical challenges further.

Recommendations for Non-technical Challenges

Regarding the non-technical challenges previously discussed, our recommendation is that the talent management and the legal aspects should be prioritized and further investigated when adopting a multi-cloud strategy. We consider the legal aspects to not necessarily be very costly to handle but they need to be addressed and resolved as they otherwise could potentially hinder the multi-cloud adoption completely. For talent management, it is important to decide in an early stage if competence should be brought in externally or developed internally. If the plan is to gain multi-cloud competence by educating current developers, a decision regarding how the competence should be distributed has to be made. One possibility is to educate one or two developers as experts of each CSP, which might be a cost-effective approach, but there is a risk of losing the competence if the developer decides to leave in the future. Another possibility is instead to educate all the developers to be able to work with all the CSPs. This would require a significantly higher initial investment but the risk of losing competence would not be as substantial. An additional advantage with this approach would be that the risk of reaching a "bottleneck" would be decreased since the competence is split more evenly and the workload could therefore also be split more evenly. For the remaining non-technical challenges, economy and sustainability, we consider them to be important in order to optimize a multi-cloud adoption but they will generally not hinder the multi-cloud adoption from being realized. To conclude, our recommendation is that the legal aspects should be addressed early in the adoption process and if possible, the current developers should be educated to work with multiple CSPs.

Challenges Addressed Going Forward

Since the literature study and the interviews agreed upon that the portability challenge is the main challenge when it comes to adopting a multi-cloud strategy we will prioritize the exploration of it over the other technical challenges. Due to the perception that interoperability is not a major challenge for the IKEA team we will prioritize it less going forward, although we still plan to investigate if there is a possibility that the IKEA team is overlooking certain aspects. As previously explained, vendor lock-in can be seen as a consequence that mainly arises when the portability and interoperability challenges get out of control. Going forward we consider that finding solutions to the originating challenges will also help us reduce the severity of the vendor lock-in challenge. However, the other aspects of vendor lock-in will also be taken into consideration. The remaining technical challenge, the security aspects, will not be given the same priority as the portability challenge but will be investigated to the extent that our limited time allows.

Chapter 4
Identification of Solutions

In order to provide the IKEA team with recommendations for how to handle the technical challenges discussed in section 3, we need to explore RQ2a and RQ2b further. The solutions for RQ2a were identified by performing a second literature study where existing research and practices were investigated. Our motivation for only performing a literature study for finding possible solutions is that we during the first literature study got the impression that this area is already greatly explored. We do not consider ourselves experienced enough to introduce a new kind of solution that has not already been discussed in previous studies. Instead we will discuss, critically analyze and possibly combine the solutions we find in this literature study in order to find suitable solutions for our purpose. We also plan to influence the solutions with our own ideas in the next chapter where we plan to further explore some of them practically with hands-on experiments. In order to answer RQ2b, a part of the literature study included studying relevant papers from the Configuration management course.

This chapter will first discuss possible solutions identified in our second literature study for each technical challenge previously identified. Then, the variant solutions are discussed in its own section as we do not necessarily consider them to be coupled to a specific challenge. This is followed by an overarching discussion where the solutions are discussed in a more general manner. The chapter is then concluded by a discussion and motivation of which subset of the solutions we plan to further explore in the next chapter.

4.1 Literature Study 2

The chosen methodology for collecting data for the possible solutions is by conducting another literature study. As previously described in section 2.2.2, a literature study will allow us to identify a broad range of possible solutions that we later can further explore and their suitability can be evaluated for our given context. Due to delays in receiving the right access level for required CSPs, we could not begin our hands-on experiments as planned. This

resulted in us spending more time doing this literature study and less time experimenting than we initially had planned for. During this literature study, 25 new articles were read and summarized and approximately 15 more were briefly read. In addition to these new articles, we also utilized some of the findings from literature study 1 where 75 articles were studied in order to identify possible solutions.

4.1.1 Portability

Based on the findings from RQ1 we consider the lack of portability to be the main challenge that needs to be addressed. In this section we discuss possible solutions identified in literature that address different aspects of the portability problem.

Cloud Agnostic Approach

Adam states in his article that a cloud agnostic development strategy is becoming a prerequisite for organizations using multiple CSPs [55]. The requirements and expectations within the industry today require both applications and cloud infrastructure to be compatible across different CSPs. A cloud agnostic development strategy means that systems and applications are developed without dependencies on a specific CSP [56]. In other words, the applications are developed to be highly portable and can be deployed on any CSP with no or minor adjustments. If a cloud agnostic strategy is fully implemented, the portability problem would be completely resolved. Even though the concepts used are not too difficult to understand, it is however much more complicated to implement in practice. Another limitation to be aware of when developing cloud agnostic is that unique features provided by a single CSP cannot be fully utilized since the agnostic approach has to be independent of CSP specific details. A cloud agnostic strategy utilizes multiple practices to realize this high level of portability. Two common approaches that are often used are abstractions and containerization. These approaches are further discussed in the following subsections.

Abstractions

Toivonen explains in her article that the use of abstractions is one solution that facilitates the usage of multiple CSPs for customers of the cloud [25]. She further describes that abstraction layers hide the differences between CSPs and let the developer neglect which CSP the application is intended to be run on. In the article by Ranabahu et al. the authors investigate if abstractions can be used to provide a more unified programming methodology for cloud computing [57]. They discovered that it in fact can be used and it can also be considered an effective and feasible solution for the intended purpose. However, Elkhatib highlights an important aspect to take into account when considering solutions for cloud portability based on abstractions [27]. Using abstractions generally entails that only the least common denominator between different CSPs can be used and unique services offered by a single CSP can no longer be utilized. Elkhatib also mentions that a large amount of research regarding abstraction in the cloud computing domain has been done which has resulted in a vast number of projects, where open source projects such as *jClouds* and *Libcloud* have gained the most attention.

We believe that Elkhatib emphasizes a critical weakness of abstractions as a solution for handling multiple CSPs. Only using the least common denominator between different CSPs will result in that no unique CSP features can be used. If an organization adopts this approach, there is a risk that competitors can offer better services by utilizing their CSP's full potential resulting in a competitive advantage for the competitor. However, a considerable advantage with the use of abstraction is that once it has been implemented, adding support for an additional CSP can be achieved for a very low cost since the codebase has already been prepared.

Containerization

Bernstein states that virtualization, a method for sharing physical resources between users and different applications, can be considered one of the pioneering concepts behind modern clouds [58]. Virtualization has generally been achieved by virtual machines (VMs), but containerization has emerged as an alternative virtualization approach that reduces overhead and therefore increases the utilization of computing resources [59]. The overhead is reduced by sharing a common host operating system (OS) instead of requiring a host OS for each VM. This allows containers to utilize the resources better. Containers can be defined as lightweight packages of application code bundled together with dependencies like different runtime versions and required libraries needed for running the software [60].

There are several advantages that come with the use of containers but the most relevant one for the portability aspect is the increased application portability. Watada et al. strengthen this reasoning by explaining that containerization has revolutionized the way the software industry is working due to its lightweight and highly portable nature [59]. Containerized applications can be moved with little or no modifications between different development environments due to the virtualization [61]. This is especially desired by companies operating in a multi-cloud environment, where containers are used for migrating applications between different CSPs [62]. However, containers mainly solve portability issues at the infrastructure level but there are still other levels and aspects of portability that containers don't necessarily solve. For example, when application code within containers is highly coupled to CSP specific services, portability will still be an issue [63]. Even at the infrastructure level, there are aspects that can lock the customer to a specific CSP, such as services for scaling and instance grouping. Containers can not solve this by themselves, and would generally require some form of abstraction or avoidance of these services [61].

Cloud Orchestration and Frameworks

Tomarchio et al. describe cloud resource orchestration frameworks (CROFs) as systems that manage complex operations such as selection and deployment of CSP resources where the objective is to guarantee a qualitative delivery of applications [64]. The support for multiple CSPs is one of the most important features for CROFs since it allows the customers to take full advantage of the offerings from the CSPs and choose the combination of services that fit their needs the best. Tomarchio et al. further state that multi-cloud computing presumes that there is no agreement of services between the CSPs and a third party cloud broker is responsible for providing the CROF [64]. The main goal of these cloud brokers is to ease the

portability of applications among their supported CSPs [22][43][65].

The advantage of using CROFs or cloud brokers is that they address and solve a big part of the portability between CSPs. However, this also implies that you have to pay for their third party services on top of the costs of the CSPs' services. Another aspect to consider is that while this allows you to develop applications for multiple CSPs, you are now locking your application to the CROF or cloud broker instead of the CSP. This implies that you can only use the CSP services that the cloud broker supports which might lock you out of some services [66].

Another type of framework offered by third parties are code transformation frameworks. Similar to the use of cloud brokers, applications can be developed without a specific CSP in mind. This type of framework instead uses some kind of transformation engine that together with a transformation plan changes the code to work for a specific CSP [38]. This type of solution does however come with the same drawbacks as the cloud brokers. Code transformation is only possible for the CSPs and services supported by the framework, which limits their usability.

Another type of orchestration tools are infrastructure as code (IaC) tools. Morris describes in his book that IaC is an approach for automating infrastructure based on common practices from software development [67]. He further explains that IaC is a concept for making consistent applications, repeatable common routines for provisioning and system changes as well as updating configurations in a uniform way. Deo states that the benefit of using IaC tools is that it provides an efficient software development life cycle [68]. They give the developer an opportunity to provision infrastructure by using scripts that can be run multiple times and in a later stage even be used for automatisation. Brikman argues in his article that Terraform is the preferred IaC tool when it comes to provisioning cloud resources from multiple CSPs [69]. Brikman elaborates his reasoning in his book where he argues that Terraform provides several advantages over other IaC tools such as having an immutable infrastructure, a declarative language and a huge community [70]. We consider Terraform to be a suitable tool for developers to get a unified way of interacting with resources from different CSPs. If Terraform is adopted, not only are the possibilities for automatisation much higher, each developer no longer needs to learn and interact with the proprietary cloud consoles of each CSP.

4.1.2 Interoperability

Even though the IKEA team does not consider interoperability to be a major challenge for handling multiple CSPs, we still consider it important to discuss how this challenge is addressed in the literature. This will help us determine if the IKEA team already has implemented practices that solve this problem or if the severity of it is underestimated.

Standardization

As previously discussed in section 3.1.1, the interoperability challenge mainly arises due to the lack of agreed-upon standards between CSPs. In the article by Ranjan, he states that

the solution for the interoperability challenge between CSPs is to introduce and agree upon standards, something that is currently under development [32]. However, this has been something that has been under development for a considerable amount of time and there have been several standards proposed so far but none of them have been widely adopted [64]. Other researchers like Opara-Martins et al. state that there is a low probability of a widespread adoption of standards ever occurring within the cloud computing industry [12]. We agree with Opara-Martins et al. and consider it highly unlikely that a widespread adoption of standards for the current services happens any time soon. It is not in the interest of the CSPs to introduce standards that would facilitate the communication with other CSPs as they want their customers to utilize only their services for all their needs. We do however consider there to be possibilities for more standardized services when new technologies are introduced by other actors than the CSPs themselves. If they provide a well-defined standardized way of using their services, each CSP would have to support the new technology in order to remain competitive which would facilitate interoperability between CSPs. An example of this is the microservice architecture which we will further discuss in the next section.

Microservices

Thönes defines a microservice as a small application that can be deployed, scaled and tested independently with only a single area of responsibility [71]. Due to the single area of responsibility, several microservices are combined to form a complete application or service. Dragoni et al. explain that microservices are only communicating with each other through their published interfaces and hence, the interoperability between the services is vital for their success [72]. In order to address these interoperability concerns, the microservice architecture relies on standardized communication protocols such as REST APIs. In contrast to cloud computing in general where there are no widely adopted standards, the microservice architecture has clearly defined interfaces, data formats and protocols [42]. This allows each microservice to be developed in isolation and only the agreed-upon standards need to be taken into account when the services are later exposed to others. This also means that microservices can be deployed on different CSPs and still work as a complete distributed application [73]. In other words, if a microservice architecture is adopted, the interoperability concerns are addressed by the architecture itself and should therefore not be as severe when adopting a multi-cloud strategy. It is important to be aware that a microservice architecture is not always ideal for every application and there is a cost associated with maintaining the decoupling which has to be considered.

Third Party Frameworks

Similar to the portability problem, there exist several different CROFs that also address the interoperability problem. Frameworks such as *mOSAIC* and *cloud4SOA*, which target semantic interoperability issues between CSPs, might be suitable alternatives but the previously discussed issue of being limited to only the services supported by the frameworks still exists [64]. Another disadvantage with different CROFs that we have discovered when researching the area is that almost none of the mentioned CROFs seem to have any form of up-to-date version available and generally lack maintenance or further development.

Even though there are several disadvantages with the use of CROFs, we still consider the underlying ideas and use of abstractions very well motivated and have potential to both solve the portability and interoperability problem. While the lack of support makes these frameworks not viable alternatives for organizations like IKEA, we think that they can be used as inspiration when we further explore abstractions as a solution practically in the next chapter.

4.1.3 Vendor Lock-in

In many aspects, the vendor lock-in problem is a direct consequence of the lack of portability and interoperability. If these challenges are solved, we would be able to move applications across different CSPs and interconnect applications hosted on different CSPs. This would mean that we are free to choose which vendor we would like to use, only limited by our own knowledge and CSP competence. The technical aspects of the vendor lock-in problem would in this sense be resolved. However, the vendor lock-in problem can also be alleviated in other ways, such as a cost-benefit-approach during decision making and having a well-defined exit strategy. These approaches will be discussed in the following subsection.

Develop a Clear Exit Strategy

In the article by Opara-Martins et al. the authors compare the use of CSPs with a form of outsourcing [12]. They discuss that cloud computing essentially means that applications and sensitive data is moved beyond the corporate firewall, and the same precautions as any other form of outsourcing has to be taken. The Cloud Security Alliance explains that the golden rule of outsourcing is to develop a clear plan for how to exit the contract [74]. Opara-Martins et al. agree with this statement and discuss the importance of having a well-defined strategy for how to exit a contract in order to avoid vendor lock-in from occurring.

Hosken defines an exit strategy as an organizational plan developed for ensuring that CSPs can be replaced or replicated efficiently without any major disruptions and help corporations to better respond to changing market opportunities [75]. He discusses the importance of developing the exit strategy prior to the CSP adoption since an exit strategy developed afterwards would require each migration decision already made to be revisited.

We agree with all of the authors above that emphasizes the importance of having a well-defined exit strategy. By considering and developing an exit strategy in an early stage, several of the pitfalls that reduce portability can be avoided. By having the exit strategy introduced early in the cloud development process, the exit strategy can influence the application design and therefore lock-in situations could be more easily avoided.

Avoid Replacing one Lock-in with Another

As previously discussed in section 3.1.1, Hohpe reasons in his article that escaping a vendor lock situation generally results in the introduction of another lock-in [46]. For example, avoiding a vendor lock-in by adopting an open-source solution instead of proprietary one will result in less lock-in to the provider but there is now instead another lock-in to the open-source product. Hohpe also elaborates his thoughts further by discussing that avoiding

lock-ins also result in lock-out from unique desirable features [46]. This means that avoiding vendor lock-in completely is not necessarily the best solution. Hohpe instead argues that the most important thing is to make conscious decisions and compare the benefits and costs. By taking the switching cost aspect and the unique utility aspect into consideration he introduces the 2x2 matrix illustrated below in figure 4.1 which aims to provide guidance regarding if a lock-in situation is worth the cost or not.

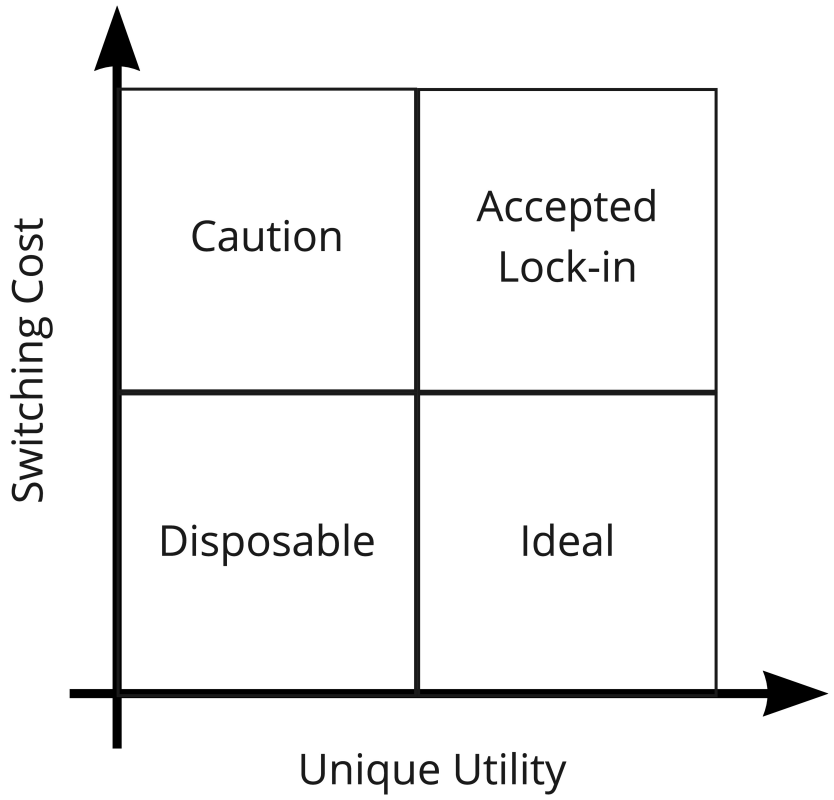

Figure 4.1: The vendor lock-in matrix by Hohpe [46].

We agree with Hohpe that a vendor lock-in does not necessarily have to be a negative thing as long as the pay-off is greater than the cost. With the vendor lock-in matrix illustrated in figure 4.1 in mind, the ideal quadrant means that the cost of switching is low whilst the unique utility offered by the service is very high. However, these ideal lock-ins are very rare as unique utility often comes with a price in the form of lock-in. Therefore, we also consider a situation with a high switching cost to be an acceptable choice as long as a unique utility or competitive advantage is gained.

For organizations the vendor lock-in problem is very difficult to completely avoid whilst remaining a competitive alternative on the market. However, the negative effects of vendor lock-in can be mitigated by making conscious decisions and comparing benefits and costs of the lock-in situation. In some cases, avoiding one lock-in can result in a better lock-in with lower switching costs while still offering similar utility. In other cases the switching cost might remain the same while the unique utility is lost and hence, the avoidance is not desirable.

4.1.4 Security Aspects

In cloud computing, security is a fundamental aspect that has to be considered by any organization adopting it. As discussed in section 3.1.1, in a multi-cloud context the security aspects tend to get further complicated and complex which is something that has to be addressed. Therefore, in this section we will discuss possible solutions to the different security aspects and how they should be handled in a multi-cloud context.

Finding Vulnerabilities

Aceto et al. state that suitable monitoring systems are needed for handling the security concerns in cloud development [76]. Norman agrees with this statement but does also emphasize how monitoring can be used for alleviating security risks by introducing a tool for alerting and analyzing long-term trends and current applications [77]. While CSPs generally provide these tools for their own platforms, we consider third party solutions such as Prisma Cloud to be the ideal solution for monitoring applications and scanning for vulnerabilities across different CSPs in a unified way [78]. By doing the monitoring in a unified way, there is less dependence on a specific CSP technology and the adopted solution will work identically independent of which CSP is adopted.

Secret Management

Similar to the monitoring concerns, Walsh also discusses the need of handling secrets such as access tokens in a unified way across different CSPs [79]. He elaborates his reasoning by explaining that the sensitive data that was previously stored on premise is now hosted in the cloud which is something attackers find very appealing. He does also address the secret management concern regarding how to guarantee that secrets are in sync across all the different CSPs used. We believe that he describes a scenario very closely related to the double maintenance problem where secrets need to be kept in sync across CSPs [17]. For this reason, we consider a third party solution that functions across different CSPs, like HashiCorp Vault, to be a suitable solution for organizations using multiple CSPs [80].

Access Management

A unified and centralized access control system would improve efficiency for multi-cloud applications and reduce the risk of developers lacking the right permissions to efficiently do their job. However, this kind of approach would result in developers generally having more access than they need which would increase the severity of a situation in which an account is compromised [81]. When it comes to access management it is generally recommended to give the least amount of privileges needed for the developer to do their work [82]. In the case of multi-cloud this would mean that access is handled for each CSP and not in a unified way in order to minimize the impact a compromised account could have. However, an approach that we consider improving the access management is by making the process of requesting access to a CSP more uniform inside the organization. This will result in less confusion for the developer but not result in any excess access being given. By easing the process of requesting access, the access control could be more fine grained without hindering the development.

4.2 Variant Handling

After the challenges had been identified in chapter 3, we are now confident that support for handling multiple CSPs will result in variants of the same application to some extent. In order to use the CSP specific services there needs to be some parts that are unique for each specific CSP. This implies that the parts of the application that involve the CSP will need to be handled differently depending on which CSP is used. Multiple copies where some parts are identical and other parts are different are variants of each other. We do however still need to figure out the best approach for handling these variants. This section will therefore discuss two solutions from the configuration management literature as well as an alternative solution found in the literature study.

4.2.1 Single Source Variants or Variant Segregation

Mahler discusses two different approaches for representing variant components, single source variants and variant segregation [18]. As we have previously discussed in section 2.3.4, single source variants is a method where all variants are contained in a single source object and the variants can be extracted when needed. Variant segregation can on the other hand be described as maintaining a separate copy for each variant of a specific component. Single source variants are preferred when the amount of difference between the variants is small and can be isolated whilst variant segregation is preferred when there is a greater difference between the variants.

In a multi-cloud context, our initial thought was that the common parts should be a large percentage of the code as the application specific parts are all the same and only the CSP specific parts should differ. According to Mahler's recommendations this would mean that a single source variant approach is preferred [18]. However, Mahler also states that the number of dimensions the variants differ in is also an important aspect, and in the case of multi-cloud, the only aspect of difference is the CSPs. This means that variant segregation can be used ideally by first localizing the CSP specific parts followed by segregating them into variants. At this point, the variant would have next to no common code between them, hence this approach would also greatly reduce the threat of the double maintenance problem. Since changes to a specific variant would only affect CSP specific parts, there would not be any risk of the same changes having to be made across variants and the double maintenance problem will therefore be avoided [17].

4.2.2 Feature Models in Software Product Lines

Clements define product lines as a set of products that address a specific market segment or fulfill a particular need [83]. More specially, software product lines (SPL) are defined by Nešić et al. as portfolios of system variants in an application domain [84]. SPLs are used to utilize commonalities and handle the differences between variants to increase systematic software reuse, increase maintainability and reduce costs [85][86]. Due to the complexity that SPLs generally result in for the codebase, feature models are generally introduced to manage and model the commonalities and variations of the different variants. Nešić et al. explain that

feature models organize features and can be used as a means of communication for maintaining an overall understanding of a system by modeling commonalities and differences [84]. Cavalcante et al. discuss that SPLs and feature models could become a powerful concept in cloud computing where commonalities and variabilities could be used for modeling the services from the CSPs [87]. They also introduce a model for how SPLs and feature models can be used to compare different CSPs. By introducing attributes to the feature models, aspects such as pricing, availability and quality of service parameters can be included and be used to facilitate the comparison of the CSPs' services.

While we understand the benefits that follow with SPLs and feature models, we want to argue that they introduce additional objects that have to be maintained and kept up-to-date. The additional work required for the feature models can easily outweigh the benefits of getting a better overview and understanding of the system as a whole. Unless the feature models are automatically generated and updated, we consider them to not be an ideal solution for maintaining variations of multiple CSPs. In cloud computing, we agree with Cavalcante et al. that the main usability of feature models is more related to the comparison of CSPs rather than maintenance of the variants.

4.3 Discussion

After we have compiled all the possible solutions, we discovered that abstractions seem to be a part of several other solutions. Several of the discussed CROFs use abstractions for increasing both the portability and the interoperability aspects. As previously discussed, abstractions and containerization are both important practices for realizing a cloud agnostic approach. A cloud agnostic approach would solve the issues related to the portability and the interoperability aspects and hence also alleviate the vendor lock-in problem. Furthermore, abstractions are also a substantial part of the variant handling solutions. However, even though the use of abstractions seem to solve many of the problems related to handling multiple CSPs it is important to realize that they come with a high initial cost. The required development time is greater for implementing abstractions instead of using the CSP specific services with no further precautions. An additional aspect of introducing abstractions is the increased maintainability that follows. Although the initial implementation might require extra development time, further support for more CSPs has a much lower cost as the codebase is already prepared. Due to the frequent occurrence of abstractions in different solutions and the cost associated with implementing them, we consider it important to explore them further.

As previously discussed in section 3.2.1, the developers of the IKEA team consider interoperability to not be a major issue for them. After we had conducted the literature study for finding solutions, we discovered that microservices seem to be a solution to the interoperability problem. We also know that the IKEA team is using microservices for a lot of their applications and it is therefore a possible reason why they do not consider interoperability to be a problem. However, we still consider data synchronization to be an aspect of interoperability that the IKEA team underestimates the severity of. We have concluded that standardization is not to be expected and due to semantic differences between the CSPs, this

is something that could impose a threat to the IKEA team's intention of using another CSP as backup.

4.4 Delimitations

In the next chapter we will further explore some of the solutions identified by conducting practical hands-on experiments in order to explore how the theoretical solutions are translated in a practical context. By further exploring the solutions, we expect to gain insights regarding how easy the solutions are to implement and what their corresponding advantages and disadvantages are. However, due to the scope of the thesis, we do not have enough time nor resources for investigating all the solutions identified. In this section we will therefore discuss and motivate which solutions we decided to further explore.

Considering the importance of the portability challenge we feel a need to further explore solutions that address this problem. We have discovered that IaC and more specifically Terraform seems to be a good practice for provisioning resources from multiple CSPs. As it also enables automation, which is desired by the IKEA team, we have decided to further explore the use of Terraform in the next chapter. Due to the lack of support for the other identified CROFs we have decided to not explore any of these further in practice. However, the concepts that these frameworks build upon, especially the ideas regarding how they use abstractions, will be taken into consideration when further exploring how abstraction can be used to solve the portability problem. The ultimate goal would rather be to explore how feasible it is to implement a fully cloud agnostic application.

When it comes to interoperability, we feel that we have highlighted some important potential solutions for the problem that could help the IKEA team validate their previous perception of interoperability not being an issue for their main multi-cloud ambitions. However, we feel that exploring these solutions further will not contribute with any major new insight and that these solutions are better suited for the IKEA team themselves to explore further.

The solutions for the vendor lock-in problem that we have presented are also something that the IKEA team themselves should further explore. Practical hands-on experiments would not contribute to more insights regarding how these potential solutions could alleviate the vendor lock-in problem for the IKEA team. As previously discussed, we consider further exploring the portability solutions, and therefore indirectly the vendor lock-in problem, to be a more effective use of our time and resources.

Our recommendations for how to handle the security aspects are given in section 4.1.4. In short, we consider third party solutions to be the best way to address monitoring, vulnerability scanning and secret management in a multi-cloud context. The access management should however not be handled in a unified manner since access privileges should generally follow the least privilege principle. Apart from these recommendations, we do not plan to further explore the security aspects in our hands-on experiments.

As discussed in section 4.2, we concluded that variant segregation seems to be the best way

to increase maintainability of an application in a multi-cloud context. We have decided to explore this approach in practice and to not investigate the single-source approach further. SPLs and feature models could be a solution that increases the maintainability of variants in a multi-cloud context. But as of now, we are under the impression that more research about the concepts is needed. If we were to explore this approach ourselves, it would require more time and resources than we have available. We have therefore decided to not further investigate it but encourage future research to explore the topic in the context of multi-cloud further since we have not been able to find any research about it.

Chapter 5
Practical Exploration of the Solutions

The objective of this chapter is to explore the theoretical solutions identified in the previous chapter in a practical context by conducting hands-on experiments. Our motivation for why we consider hands-on experiments to be a suitable approach is that we consider it to enhance our theoretical knowledge with practical experience and provide us with an opportunity to get a deeper understanding of the possible difficulties encountered when implementing the theoretical solutions in practice. By gaining this experience, we will be able to provide the IKEA team with higher quality and better motivated recommendations for their ambition of handling multiple CSPs. We will also be able to identify which parts that should be trivial for the IKEA team and which parts that might require more work and future research. Conducting hands-on experiments will also allow us to determine if a proof of concept can be produced and it will therefore allow us to conclude RQ3.

This chapter will first introduce how we conducted our hands-on experiments and the different stages they included. We will then conclude the chapter with a discussion of our overall findings and the results we obtained from the experiments.

5.1 Hands-on Experiments

The purpose of this section is to discuss our process of creating a small application that is vendor-locked to an initial CSP, which in our case was GCP. The application will then be made cloud agnostic and migrated to a second CSP, in our case the Chinese CSP Alicloud. We will first discuss the context behind the initial application and how it is locked to GCP. Thereafter, we will discuss the migration process to Alicloud and the solutions we adopted to make it cloud agnostic. Finally, we will discuss how we investigated the impact of the changes by adding support for a third CSP, Azure.

5.1.1 Initial GCP Experience

In order to gain practical experience, our supervisor from the IKEA team advised us to complete the pic-a-daily lab series about GCP [8]. He considered it a suitable start as several of the services explored in the lab series are services that the IKEA team is using themselves in their daily development. We followed the advice and after we had completed it we considered ourselves to have a good perception about the GCP services and their use cases. This training contributed to our thesis project by allowing us to spend more time exploring the solutions instead of having the exploration process hindered by not knowing the fundamentals of GCP. Since we in section 4.1.1 discovered that IaC tools are suitable for handling resources across multiple CSPs we considered it to be a good idea to get experience working with it and explore how it could contribute to our thesis work. We therefore decided to repeat the lab series again but instead of using GCP's cloud console, we used Terraform. By using Terraform we discovered that provisioning, changing and deleting cloud resources were much easier and more manageable once the initial Terraform script had been implemented. This was not only helpful during the experiments but we also consider it to improve the maintainability and in general facilitate the entire process of working with cloud resources.

The GCP Project

The objective of the hands-on experiments is to further explore the solutions in an attempt to make the application able to be deployed on multiple CSPs. However, the application developed in the pic-a-daily lab series ended up including a wide range of GCP services and migrating this application to other CSPs would require too much time and resources to fit the scope of the thesis. We therefore decided to scale down the initial pic-a-daily application to only include three different processes: download a picture from a bucket (a cloud storage container), analyze the color scheme of the picture and store the result in a database. These three processes allowed us to explore GCP services that the IKEA team themselves use in their applications, such as Cloud Storage, Cloud Functions and Firestore. Cloud Storage is GCP's service for storing binary large objects in buckets. Cloud Functions is GCP's FaaS that allows the user to run small code snippets in the cloud. Firestore is GCP's NoSQL database where data can be stored structured as documents and collections. The application itself also used Google's Vision API to analyze the picture uploaded to the bucket [88].

At this point, we had a small fully functional application running on GCP. However, the application was very vendor locked to GCP on both application level and cloud infrastructure level. The cloud infrastructure level describes the different components required for utilizing cloud computing, including the hardware, abstracted resources, storages and other network resources [89]. Furthermore, as the application was inspired by the pic-a-daily lab series, no initiatives had been taken to reduce the vendor lock-in to GCP, rather the opposite since the lab series only introduced several GCP unique features.

5.1.2 Migration to Alicloud

As discussed in section 4.1.1, adopting a cloud agnostic implementation would result in the highest possible portability of the application. Even though the objective at this point was

to mainly get the application up and running on Alicloud, we also wanted to explore if we could make the application cloud agnostic or at least reduce the vendor lock by preparing support for any other CSP. This approach would require us to remove the vendor lock-in on both the application level and the infrastructure level.

The Application Level

Our first step was to investigate lock-ins on the application level where we quickly realized that the Vision API provided by Google was causing a vendor lock. While this API technically could be used in applications hosted on other CSPs than GCP, it was not usable on Alicloud as all Google services are blocked in China [2]. This is a small takeaway that organizations that want to adopt a multi-cloud strategy must be aware of. To solve this problem we were required to find another API, with similar functionality, that could replace the Vision API. Even though we put an effort into finding an appropriate replacement API, we did not manage to find one that offered all the functionality that the Vision API did. This can be seen as an example of a vendor lock-out situation where we had to accept less functionality and more restricted use cases in order to escape the vendor lock-in. The API we decided to use was the open source image-analyser tool OpenCV which, similar to the Vision API, could analyze pictures and find the most frequent occurring colors in them. However, the more advanced features such as revealing the image characteristics were no longer available [90].

The next step was to find the corresponding Alicloud services to Google's Cloud Storage, the service the application was downloading the picture from, and Firestore, the database where the resulting color scheme was saved to. We were under the impression that corresponding services should be available across all the major CSPs since the services we used and the functionality they offered can be considered commodities. Although we managed to find suitable corresponding services in Alicloud, Object Storage Service and Tablestore, we encountered another problem related to the semantics of the services. As pointed out by the literature and discussed in section 3.1.1, corresponding services from different CSPs can have semantic differences in how they are used. An example of this is illustrated in the figures below where the color scheme of the picture is saved to a database on GCP and Alicloud respectively.

```python
def add_colors_to_database(blob_name, colors):
    """
    Add the colors to the firestore database
    """
    # Connect to Firestore
    db = firestore.Client()

    # Get the reference to the Firestore document
    doc_ref = db.collection(u'pictures').document(blob_name)

    # Create and add the new document to the firestore collection
    doc_ref.set({
        'created': datetime.now(),
        'color1': colors[0],
        'color2': colors[1],
        'color3': colors[2]},
        merge=True)
```

Figure 5.1: A python code snippet illustrating how picture data is uploaded to Firestore.

```python
def add_colors_to_database(blob_name, colors):
    """
    Add the result to the tablestore database
    """

    # Connect to Tablestore
    db = tablestore.OTSClient(
        'https://Cloud-agnostic.cn-hangzhou.ots.aliyuncs.com',
        access_key,
        access_secret,
        'cloud-agnostic-t',
        retry_policy=None)

    # Predefined table name
    table_name = 'pictures'

    # Create the Tablestore attribute columns
    attribute_columns = [
        ('created', str(datetime.now())),
        ('color1', colors[0]),
        ('color2', colors[1]),
        ('color3', colors[2])]

    # Predefined primary key
    primary_key = [('name', blob_name)]

    # Create the new tablestore row to be added
    row = tablestore.Row(primary_key, attribute_columns)

    # Required for put_row
    condition = tablestore.Condition(tablestore.RowExistenceExpectation.EXPECT_NOT_EXIST)

    # Add the newly created row to the database
    db.put_row(table_name, row, condition)
```

Figure 5.2: A python code snippet illustrating how picture data is uploaded to Tablestore.

As figure 5.1 and figure 5.2 illustrate, the process of saving the picture color scheme to a database on respective CSP is different from each other. For example, Firestore is structured using collections and documents while Tablestore uses rows and columns with primary keys even though both are described as NoSQL databases. This example might not be considered a severe challenge and an experienced developer will most likely not have any issues figuring

out how they both work. However, the differences impose a possible threat related to the mental lock-in scenario as previously discussed in section 3.1.1. Mental lock-in occurs when developers make assumptions based on their previous knowledge. In this case, making assumptions based on how GCP and Firestore work when developing an Alicloud application using Tablestore would cause errors. Very few of the methods for working with the databases are directly translated between GCP and Alicloud, it is also not guaranteed that the workflow follows the same order in both of them. When working with two separate CSPs it is also important to be aware of the threat related to developing and maintaining both of them over time. Since there is a distinct separation between them and due to incompatible functionality the developers must be fully aware of which CSP they are currently working on and they have to be able to switch back and forth between the two different mindsets required for developing with each CSP.

An example of when the mental lock-in affected us during the thesis was when we tried to extract the name of the image that was supposed to be downloaded from the CSP. As the function was supposed to be run when an image was uploaded to the storage service of the CSP, we created a trigger that reacted on an upload. This trigger executed the function and had an event containing all the information related to the upload. We discovered that the process of extracting the image name from the event vastly differed between GCP and Alicloud even though they shared the exact same name and served the same purpose. As illustrated in figure 5.3 below, the process of extracting the name from the GCP event was simple. This affected our assumption regarding how the extraction process worked on Alicloud as we expected it to follow a similar simple structure. An important note on this subject is that almost all detailed documentation of the Alicloud services is given in Chinese and finding the information in English is not trivial. In an attempt to avoid spending too much time searching for documentation, we assumed that we could figure out how to extract the image name through trial and error. This ended up not being as trivial as we first expected and finding the image name ended up being a difficult process. This was due to the event on Alicloud having a completely different structure compared to the event on GCP. The way of correctly extracting the image name can be seen in figure 5.4 below.

```
10    # Get the image name
11    img_name = event['name']
```

Figure 5.3: Code snippet illustrating the process of extracting the image name on GCP.

```
22    # Get the image name
23    img_name = event['events'][0]['oss']['object']['key']
```

Figure 5.4: Code snippet illustrating the process of extracting the image name on Alicloud.

Since the two solutions were very different from each other we could easily identify the problem in an early stage since several error messages were thrown at us. However, the differences that are most threatening are generally the opposite where only minor details differ and no

error messages are produced. In these cases it is very complicated for the developer to notice the errors and they can easily pass unnoticed. The code might even compile and initially work as intended. This can result in small behavioral errors that can go unnoticed for long periods of time and once noticed, finding the root cause of the errors can be next to impossible.

Abstractions and Variant Segregation

Initially, there was no distinct separation between the GCP specific code and the Alicloud specific code. The main file simply chose which CSP code to use based on an if-statement. However, as previously mentioned no Google services can be run in China. We also discovered that not even imports of Google services are allowed even though they are never used. We saw this as an excellent opportunity to use suggested theoretical solutions from chapter 4, abstractions and variant segregation, for solving this issue. The solution we thought was best suited for solving the problems we were facing was to isolate the CSP specific parts of the code. We then created two abstract methods, one for downloading the picture to be analyzed from the storage service and another one for uploading the resulting color scheme to the database. This allowed us to create CSP specific variants, according to the variant segregation method, which only implemented these two methods and had no common parts between them. This approach also made the main file of the application cloud agnostic in terms of containing no CSP specific parts at all and adding support for more CSPs would hopefully not require any changes to the main file. Instead, adding support for additional CSPs would only require that a new CSP variant was created and that the abstract methods were implemented. An illustration of how abstraction and variant segregation can be combined is illustrated in figure 5.5 below.

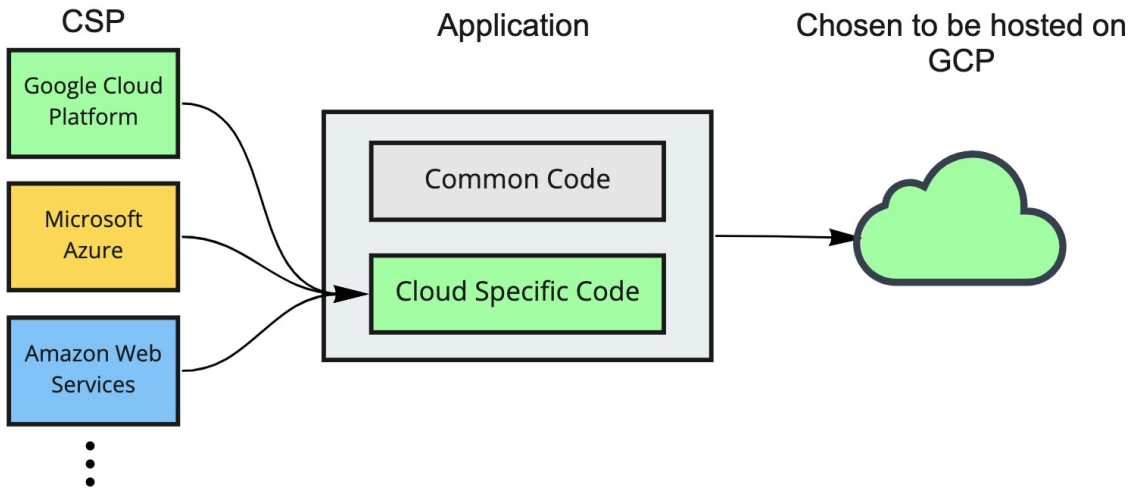

Figure 5.5: An illustration of how CSP specific variants can be abstracted out from the common parts of the code. The needed CSP variant can then be added according to which CSP the application is intended to be hosted on.

This approach led us to another challenge related to how the import of the variants should be handled by the main file and remain cloud agnostic. As the code is supposed to be run on a specific CSP each time, we considered it necessary to only import the required CSP variant. Since we also wanted to keep the main file cloud agnostic and not have to be updated when support for additional CSPs was added, we implemented a solution that imported the correct CSP part based on a configuration file. This means that if additional support for other CSPs is added in the future the main file does not have to be updated. Apart from the CSP specific code that has to be implemented, only a line in the configuration file has to be changed. Our implementation is illustrated in figure 5.6 below.

```
8  # Read the yaml file config.yaml and create the variable csp_part with value from the yaml file
9  with open('config.yaml', 'r') as ymlfile:
10     cfg = yaml.safe_load(ymlfile)
11 csp_part = cfg['csp_part']
12
13 # Import the csp_part as chosen_csp
14 chosen_csp = importlib.import_module(csp_part)
```

Figure 5.6: Our implementation for only importing a single CSP based on the config.yaml file.

Once the *chosen_csp* had been imported, we could use it to call the abstract methods and get the CSP specific implementation as illustrated in figure 5.7 below.

```
31      # CSP specific line to download the image associated with the event
32      blob_name = chosen_csp.get_picture(event, "/tmp/temp.png")
33
34      # Application code inbetween..
35
36      # CSP specific line to upload the result to the database
37      chosen_csp.add_colors_to_database(blob_name, colors)
```

Figure 5.7: Code snippet illustrating how the abstract methods are called from each CSP.

The Infrastructure Level

The purpose of this section is to discuss the challenges related to the infrastructure level we encountered when performing the hands-on experimenting. We will discuss how Terraform affected the provisioning of cloud resources and also how dependencies were handled differently between the CSPs.

Terraform When we first started exploring Alicloud we used its cloud console for finding and provisioning the corresponding services we needed. This quickly became a difficult process, partly due to the language differences in some parts of the cloud console but also partly because the functionality differed from GCP's cloud console, which we now can reflect on as another mental lock-in from our side. This mental lock-in situation can be considered a short-term lock-in that arises in the transition phase when adopting a new CSP and will eventually fade away as the experience working with the CSP increases. The opposite, more threatening lock-in is the long-term lock-in that is ever present when maintaining, bug fixing

and further developing code for multiple CSPs.

We saw the need for a unified way to handle the resources from multiple CSPs. As previously described in section 4.1.1, the literature study revealed that IaC tools can be used for provisioning, changing and deleting cloud resources in a unified way across supported CSPs. Therefore, we decided to adopt Terraform as our IaC tool of choice since it was highly regarded in the literature studied and is a tool that the IKEA team has already started to explore. Terraform allowed us to provision resources from both GCP and Alicloud by simply writing HCL code, the language which Terraform is using, instead of manually having to provide them in their respective cloud consoles. Like any other programming language, learning Terraform and implementing the first solution for GCP took some time to figure out. However, when we implemented the second solution for Alicloud the development time was greatly reduced and we consider it to be a much simpler and effective process than the alternative of manually provisioning the resources. The use of an IaC tool like Terraform also reduces the risk of mistakes caused by individual developers since every developer no longer needs to know every detail of every CSP in order to get the applications up and running. Instead the details are handled only once and could therefore either be done together or by an experienced developer of the CSP. Another advantage we discovered with Terraform was the ability to quickly provision resources on a CSP, something that aligns with the IKEA team's ambition to use additional CSPs as backups in case of downtimes on their current used CSP.

Dependencies Our initial plan was to use Cloud Run, Google's serverless container service, for hosting the application but we did not manage to find a suitable corresponding service on Alicloud. The service we found most similar was Alicloud's serverless Kubernetes service but we considered it to not fit our small application and this was therefore another case of a vendor lock-out situation. We instead had to find the least common denominator between the CSPs and decided that the most suitable solution that was available on both CSPs was the use of FaaS services. Since FaaS were also of interest for the IKEA team, it seemed like a suitable solution to further explore. The services were GCP's cloud functions and Alicloud's function compute. The main issue we faced when working with these services was that we noticed that they handle dependencies differently. This is an example of how even commodities from different CSPs can differ in how they work and are structured. Cloud functions require a list of all dependencies that are going to be used when executing the code in order for it to be able to install them. Function compute on the other hand, worked slightly differently. It instead required a zip-file containing the code and all the dependencies already installed. Due to our ambition of handling all the CSPs in an agnostic and unified approach, the requirement for handling these dependencies in different ways became a problem. The approach we decided to use for solving this problem was to create "set-up" scripts for each CSP that handled the dependencies in the required CSP specific way and produced a zip-archive with the correct file structure for the given CSP. However, fortunately enough the set-up scripts could be handled and executed by Terraform in a cloud agnostic way.

The problem with dependencies could possibly have been solved by introducing containers. Container services of the CSPs are much more standardized and in general only require a container image which contains all the dependencies. This would mean that dependencies could be handled in the same way for both CSPs and therefore avoiding the dependency

challenge we encountered with the FaaS services. As previously described, our ambition was to fully explore this potential solution but due to the lack of time and resources of the thesis we did not manage to do so. Instead we prioritized the work which is described in the next section.

5.1.3 Adding Support for any Cloud

Once we had successfully managed to migrate the initial GCP application to Alicloud, in what we considered to be a cloud agnostic way, we wanted to validate if our changes had facilitated the process of adding support for another CSP. We therefore decided to implement support for Azure.

Due to access management problems this part of the experiment was conducted without having access to Azure until the very end of the experiment. Instead, we were preparing the solution by implementing Terraform code although it could not be executed and tested without the proper access. However, since we were already familiar with Terraform at this stage and due to Terraform's way of handling resources from different CSPs in a unified way, we could write code that worked after only a few minor adjustments once we got the correct access. We consider this to be an example of how a unified solution can ease the development towards multiple CSPs.

On the application level, we also clearly saw the benefits of our changes that made the code more cloud agnostic. As expected, we did not need to apply any changes to the main file and we only had to create a new Azure specific part which implemented the abstract methods we previously defined.

We consider this to validate our initial assumption that adding support for another CSP would be a much easier process after the initial investment of development time in order to make the code cloud agnostic.

5.2 Discussion

The purpose of this section is to provide an overall discussion of the findings identified in this chapter in order to highlight the most important aspects for the IKEA team to be aware of and also discuss how the findings enable us to answer RQ3.

The theoretical solutions we identified in chapter 4 and further explored practically in this chapter all turned out to facilitate the process of handling multiple CSPs to various extent. For the application we developed, the use of abstractions and the variant segregation concepts did come with an initial cost and required us to spend time adjusting the code. However, when we implemented support for a third CSP we could take advantage of the new structure and we saved a lot of time and resources. The cloud agnostic approach has an increased initial cost, but once implemented, it greatly reduces the cost of future implementations. We also consider the new cloud agnostic structure of the application to be much easier to maintain and further develop as the CSP specific parts are clearly isolated. It becomes much clearer

where to apply changes depending on if the change affects common parts or CSP specific ones and the risk of unintentionally affecting other CSP variants is greatly reduced. However, it is also worth noting that a cloud agnostic approach is not always worth the required cost. Even for our small application it required a significant amount of rework to restructure the application. Although this is only a one time initial cost, for more extensive applications like the ones the IKEA team is using, it is important to first determine if the benefits that a cloud agnostic approach brings outweighs the cost of implementing it. If there are currently no plans to host the applications on multiple CSPs, is the initial extra investment of making it cloud agnostic really worth it? This is an important question that needs to be considered for each specific application since there is no optimal one-size-fits-all approach present.

Another important aspect to be aware of and that we also encountered in our experiment was the risk of being vendor locked-out. If a cloud agnostic approach is adopted it means that unique CSP specific services, that possibly could give a competitive advantage, cannot be used since there are no corresponding alternatives available on other CSPs. In our case this was illustrated by not being able to use the Vision API and Google's Cloud Run to host our application. Once again we consider it important to evaluate each situation individually and consider if it is worth it to refrain from some services that otherwise could give you a competitive advantage in order to have a complete cloud agnostic solution.

In other words, the parameters that need to be taken into account when considering a cloud agnostic implementation are the initial extra development cost, the likelihood of hosting the application on multiple CSPs and how important CSP unique features are for the performance of the application.

Regarding our third research question, whether or not a proof of concept with the findings from RQ2 can be implemented, we consider our small application to be a proof of concept confirming that it is possible. We have investigated how the theoretical knowledge we gained from the literature study can be applied to solve several of the issues related to handling multiple CSPs and also how it can be used to facilitate the development process. As the literature study suggested, both abstractions and variant segregation are two solutions that organizations like IKEA should be aware of once deciding to adopt a multi-cloud strategy and our proof of concept confirms that they can be practically implemented. With the hands-on experiments conducted, we now feel comfortable concluding RQ3.

A concern that we have is that our experiments can be considered small and for our case, isolating the CSP specific parts and introducing the abstractions did not impose any major issues. There is a possibility that the cost of abstractions and variant segregation scale detrimentally with the complexity of the application. We are therefore not completely certain what the costs of these two approaches are even though both the literature study and our experiments found clear benefits from using them.

Chapter 6
Discussion and Related Work

In this chapter we would like to evaluate and reflect upon the whole thesis work from an academic perspective. This will allow us to reflect on how our work process could have been improved by the knowledge we now have gained, help us determine the validity of our results and discuss the generalizability of them. Another important aspect of this chapter is to compare and discuss our findings in relation to existing research on the topic. The chapter concludes with a statement regarding potential future work and possible improvements.

6.1 Reflection on Work Process

In this section, we will reflect on our own process of work and discuss how we would have done the thesis differently if we were to conduct the work once again. However, we will also discuss the parts of the work process that we are pleased with and why we consider these parts to be suitable for the thesis work.

In general, we are satisfied with our work process. Our planned methodology worked well in practice and by conducting the literature studies, we managed to explore how previous research has addressed the problems related to handling multiple CSPs. It also provided us with valuable insights and knowledge that we used as a theoretical foundation when we explored the more practical parts later in the thesis. If we would have conducted the hands-on experiments prior to the literature studies, we would not have had the same theoretical foundation which would have resulted in the quality of the experiments being lower and the results would be more questionable. We also consider the interviews we conducted to have contributed with valuable practical knowledge that complemented the theoretical knowledge. For the time we invested in conducting them we consider the output to be satisfactory as they contributed with the knowledge based on the IKEA context. The hands-on experiments are another part of the thesis that we are satisfied with and the outcome contributed with new knowledge and experience that we could not have gained solely from the theory.

However, there are also some parts of the thesis work process that we would have done differently if we were to conduct the thesis again. In hindsight, we are under the impression that we could have better utilized the work done in literature study 1 for literature study 2 as well. While we consider it beneficial to separate the search for challenges and solutions in order to work towards answering RQ1 and RQ2 sequentially, we also noticed that most articles introduced both challenges and their take on possible solutions to them. By better categorizing and taking notes of the articles that discussed solutions during literature study 1, we could have saved a lot of time during literature study 2. At this point, we had summarized over 75 articles and we had to go back through them all to check for solutions as we knew some of them brought up relevant ideas. There were also some cases where we remembered reading about a certain solution or concept that we would have liked to further explore but we did not know which the corresponding article was. We therefore had to briefly reread a lot of the summaries and often entire articles to find the specific idea we had previously read.

In retrospect, reading 75 articles could be considered excessive and we would most likely have identified the same challenges associated with handling multiple CSPs if we had read 50 articles instead. However, we did not know that at the time of reading and we consider the extra articles to contribute with knowledge that validates the already identified challenges and provide more credibility to our conducted work.

Another aspect that we should have been aware of at an earlier stage was the process of getting access to the three different CSPs. We greatly underestimated the required time for getting the access that we needed for our hands-on experiments. Rather than taking a few days as both we and the IKEA team estimated, the accesses took more than a week to receive and the access to Alicloud took more than two weeks. This resulted in that the experiments could not be conducted when we initially planned them to be, and we had to restructure parts of the work process in order to utilize our time efficiently. For example, we spent more time on the second literature study and less time on the hands-experiments than we had initially planned for. In the case of getting access to the last CSP, Azure, the delay even resulted in us not having enough time left to fully explore the parts we initially intended to. This problem mainly occurred due to our inexperience of the process of getting this kind of access in large corporations such as IKEA. The problems could have been avoided if we had planned the experiments more thoroughly in an earlier stage or at least had decided which CSPs we were interested in further exploring.

6.2 Threats to Validity

We consider it important to highlight and discuss both how we have achieved a high level of credibility in our results but also discuss aspects that could be seen as a threat to the validity in order to determine how trustworthy our results are. This section will therefore discuss the validity aspects for the different parts of our thesis work.

We consider our literature studies and the corresponding findings from them to have a high degree of validity. In total, approximately 100 papers were thoroughly read and summarized

during the literature studies, and an additional 50 papers were briefly read and contributed with passive knowledge. Due to the extensiveness of the literature studies we consider the risk of us overlooking an important challenge or solution to be small. A wide range of different challenges and solutions were revealed and whilst some of them were unique for a specific paper, several of them occurred multiple times in different articles. Due to the limited scope and resources of our thesis we decided to only focus on the most important challenges and solutions that were discussed by multiple authors and it is therefore possible that we might have overlooked aspects that were only discussed once. However, the total number of different papers studied can be seen as an assurance that the challenges and solutions identified actually exist since they are discussed and mentioned by multiple authors and independent sources of information.

One thing that could be considered a threat to the validity is how the interviewees were selected. Our supervisor from the IKEA team recommended suitable interviewees to us and it is therefore a possibility that we did not get a representative picture of the developers experiences and knowledge from the IKEA team. However, since we were asking questions related to a specific area of development we considered it important that the interviewees were familiar with the topic and could contribute with their insights. If a more random selection process would have been used this would possibly give us a better representation of the IKEA team as a whole but there is also a considerable risk that some of the questions could not be fully answered or covered in a satisfactory way. However, the main objective was never to determine the experience of the average developer at the IKEA team. Instead, the objective was to reveal the collective knowledge and experience of the IKEA team and to collect as much practical experience and knowledge as possible to complement our literature study. For that purpose, interviewing the more experienced cloud developers was favorable in terms of knowledge gained in contrast to the time spent.

A threat to the validity of our results is that the hands-on experiments we conducted were small and did possibly not give a representational picture of the reality of the IKEA team. As previously discussed, the purpose of the experiments was to both enhance our theoretical knowledge by gaining practical experience and evaluate some of the theoretical solutions identified. Due to the scope of the thesis and limited time, we did not have the opportunity to fully investigate the solutions in an actual application that the IKEA team itself is using. Instead we had to develop our own small application and chose some of the CSP services we would like to further investigate. Even though we chose services that the IKEA team is using we did not have enough time nor resources to cover all their services.

Another threat to the validity is that we were not given the opportunity to investigate how our proposed solutions affect the development over time from a practical perspective. We have previously discussed in the thesis how maintenance is an aspect of development that often gets overlooked even though it can be considered a key aspect. How easy it is to both maintain the code and add additional features as a response to future events is something that should be of great importance since it can save resources and developer time. However, due to our limited time of the thesis we had no opportunity to investigate how the solutions we propose affect these aspects over time. It is therefore possible that some of the solutions we and the literature consider to increase maintainability and ease development might not

have the same positive effects long term.

The limited scope of the thesis did also compel us to investigate some of the challenges and solutions identified only from a theoretical perspective. For example, the security aspects of handling multiple CSPs is something that both the literature and the developers from the IKEA team highlighted but we did not investigate the proposed solutions for handling the associated challenges further. Similarly, we did not investigate the non-technical challenges related to handling multiple CSPs apart from what the existing research had found. If we had investigated the impact of these aspects further, it is possible that we would have found other recommendations or at least could have provided the IKEA team with more well-founded recommendations.

6.3 Discussion of Generalizability

The purpose of this section is to discuss how generalizable the findings of the thesis are or if they are only relevant in the context of the IKEA team. This section is mainly written for readers that are not familiar with the IKEA team and need to know how applicable our results are for their own context.

In general, we consider our results to be applicable outside the context of the IKEA team. Since the results of the thesis are highly based upon the foundation from the literature studies, the context of the IKEA team has not influenced the findings to a notable extent. The interviews with the IKEA team during the *identification of challenges*-phase were intended to confirm the challenges identified during the literature study rather than reveal new ones. Therefore, we consider the result from this phase to be highly generalizable. However, when we conducted this thesis we mainly considered large organizations such as IKEA to be our target audience. This means that we generally only explored the larger public CSPs that are present in many parts of the world. Small and medium-sized enterprises could maybe find value in using local CSPs which could result in other challenges and possibilities that were not addressed in this thesis. Based on the literature study, our perception is that these smaller CSPs are in favor of introducing standards in the industry since they need to collaborate with others in order to compete with the larger CSPs. They generally also provide services that can be considered commodities and customers therefore have very few incentives for getting locked to a CSP when there is no benefit in terms of a unique competitive advantage.

One aspect of the thesis work that the context of the IKEA team did influence was the importance and perceived severity of the interoperability challenge. The literature study concluded that interoperability together with portability were the main challenges for organizations looking to adopt a multi-cloud strategy. Since the IKEA team considered the interoperability challenge to not be severe for them, we put less emphasis on the challenge in the other phases of the thesis. We consider their microservice architecture and the fact that customer data does not need to be shared between different geographical regions to reduce the severity of the interoperability challenge for them. This might not be true for other companies and the thesis might not portrait how to handle the interoperability challenge in a generalizable way.

6.4 Related Work

In this section we will discuss four papers of previous research that we consider related to our own area of work. We will discuss how the results from our thesis align with previous work and motivate how our thesis contributes to the research area by addressing some of the limitations of these papers. This is achieved by addressing each paper beginning with a short discussion of why the paper is relevant for our work followed by a summary of its content and a discussion regarding the findings and limitations of the paper. The purpose of the summary is to shortly introduce the reader to the paper so that the following discussion can be grasped without the need to study the entire paper.

6.4.1 Mapping Cross–Cloud Systems: Challenges and Opportunities

Similarly to our own work, the paper by Elkhatib aims to summarize and clarify the current challenges that are present in cross-cloud environments, which he defines as a supercategory to multi-cloud [27]. This aligns with our own thesis as we also address challenges and best practices of the multi-cloud environment.

Summary

Due to the lack of standardization and the significant growth of cross-cloud and cloud computing in general, an increasing number of cloud architectures have evolved. Elkhatib believes there is a need to summarize the present situation of cross-cloud computing due to this significant growth in recent years. He presents a cross-cloud dictionary based on his experience from a workshop series as well as previous literature. The dictionary consists of four clearly defined subcategories to cross-cloud computing; hybrid-clouds, multi-clouds, meta-clouds and cloud federations. He motivates this categorization by looking at a set of characteristics and how they differ between the categories. Some of these characteristics are for example how similar the sub-clouds are, the level of abstraction and how responsibilities are divided between consumers and CSPs. Elkhatib also highlights a few challenges and development efforts that in the current state of cloud computing have to be addressed by the developer. These challenges are then discussed and analyzed from the different perspectives of each subcategory of cross-cloud computing. The article concludes by noting that cloud computing is an industry-driven domain and while there are some challenges that need to be accounted for, there are also opportunities for actors not linked to the largest CSPs to influence the state of cloud computing.

Discussion

Elkhatib discusses several different challenges associated with cross-cloud systems but he is not referencing any other sources of information apart from his own experience. Even though he can be seen as a reliable source and expert within the field, we consider his findings to not be as substantiated as we would have liked. We therefore consider our own literature study to complement and possibly strengthen some of his claims. It is also possible to see it the

other way around where his claims strengthen our findings. For example, he addresses the problem that organizations that want to fully utilize a CSP's unique value-added services tend to become more dependent on the CSP over time. This is something that aligns with our own findings stating that unique services result in some kind of vendor lock-in. Elkhatib also discusses how CROFs can be used to avoid these lock-ins but that their solutions rely on the least common denominator which is something we also found to be true. He argues that this can be suitable for some applications, which he motivates by the success of frameworks such as Libcloud. In our case, we did not consider these frameworks to be suitable for the IKEA team as the support for different services was very limited. This is an agreement with Elkhatib's theory of these frameworks not being suitable for all applications.

Elkhatib also highlights that the state of the cloud brokerage market is struggling and compares it to other third party vendors within other markets. He states the reason that cloud brokers struggle while other third party vendors, like airline ticketing, are successful must be the high level of expertise of the cloud customers compared to other sectors which means that cloud customers don't need help from third-party vendors. We do however not agree with this reasoning and instead our work has shown that it is more likely a matter of vendor lock-in. Being vendor locked to a third party broker can be considered a risk as you are dependent on them implementing support for new features and changes to the CSP. Third party brokers in other sectors such as airline ticketing do not result in a vendor lock-in situation for the customer and there is therefore no risk associated with using the third party vendor to get the best deal. If this was the case for cloud computing as well, we would likely have seen many more companies utilizing these brokers instead of developing their own solutions.

6.4.2 An Overview of Multi–cloud Computing

The article by Hong et al. provides an overview of the state of the industry with challenges and ongoing research in mind [47]. Since our thesis has the intention to reveal the challenges associated with multi-cloud computing we deemed this article to be highly relevant for us. By using the challenges discussed in the article as a foundation for our own research, we were able to get a broad overview of the current challenges within the domain of multi-cloud computing and were later able to further explore them.

Summary

In the vast field of multi-cloud computing there are currently several issues, such as vendor lock-in and security between CSPs, that have not been properly addressed. Hong et al. intend to explain the current situation of cloud computing and review current issues within the industry and how they possibly could be handled. The authors begin by explaining some fundamental concepts of cloud computing to establish a common ground for the reader that will be used to discuss more advanced concepts later in the paper. Once that has been achieved, the challenges of cloud computing are presented which have been gathered through extensive literature reviews. These challenges have then been categorized into six different categories and are further discussed according to their severity and in which scenarios they arise. The paper then shifts its focus to describe different multi-cloud solutions and how they address the previously described challenges in cloud computing. Different kinds of cloud computing

models are analyzed and under which circumstances they are an appropriate alternative is also explained. In other words, the intention of the paper is to first gather challenges related to cloud computing and multi-cloud computing and inform the reader about their presence and severity before discussing different ways of handling them in an efficient manner as possible in a multi-cloud setting. The paper is then concluded by mentioning which direction the research of the industry is heading and which areas within the domain of cloud computing that need to be prioritized by future research.

Discussion

Hong et al. present a picture where they argue that since cross-cloud platforms rely on the communication between cloud components from different CSPs, this will generate new methods for facilitating interoperability naturally. We do however think that this description presents an ill-nuanced version of the reality since they do not take the perspective of the CSPs into account. We have discovered that the CSPs are not actively working for or even striving towards finding a unified way of achieving a smooth cross cloud communication process, rather the opposite. As we have found in our literature study, it is not in the interest of the major CSPs to provide opportunities for switching to other CSPs effortlessly since it would benefit competitors rather than their own business. This is a perspective that is currently missing in the article and is something that we consider to be a threat to the validity of the results.

Our challenge identification process revealed several challenges that align with the findings from Hong et al. Similarly, we both found interoperability, portability, legal concerns, vendor lock-in and security to be common important challenges to consider when adopting a multi-cloud strategy. The major difference between how the authors and ourselves have explored the challenges associated with handling multiple CSPs is that we introduced more aspects than solely theoretical ones. As previously mentioned, we did conduct interviews with developers at the IKEA team and also performed our own experiments to further explore the challenges in an attempt to discover aspects that can't be found solely by reading. We consider our thesis work to complement the authors' findings and provide an improved overview of the current situation of multi-cloud computing.

6.4.3 Critical Review of Vendor Lock-in and Its Impact on Adoption of Cloud Computing

Even though the title of the article indicates that vendor lock-in is going to be the major area of discussion this is not the case. Instead, Opara-Martins et al. mainly discuss how the concepts of interoperability and portability relate to vendor lock-in in a cloud computing domain [12]. Our own research found that both interoperability and portability are the main challenges for a multi-cloud context and therefore we consider this article to be highly relevant for our work.

Summary

Cloud computing is still a relatively new phenomenon within the software development industry and the possibilities that it offers attracts several different businesses of various kinds. However, due to its recent establishment in the industry it suffers from the lack of well-defined standards. Rather each provider seems to offer their own unique solution with no intention of making it functional across different CSPs. As a result, once customers have chosen their initial CSP they are often vendor locked to its services with no possibility of utilizing other existing alternatives. Therefore, the authors of the article are investigating how vendor lock-in affects corporate cloud computing applications and services in order to provide a deeper understanding about the subject for future research. The authors are achieving this by conducting a review of articles and work related to the topic they are investigating. In this review, it becomes clear that both interoperability and portability are two key concepts tightly associated with vendor lock-in and the lack of either of them often result in some form of vendor lock-in occurring. It is then later discussed that standards can be introduced as a way of mitigating the risks of vendor lock-in but also that CSPs are not actively working on introducing them since it would give the customers more possibilities of switching to alternative solutions and hence risking losing customers. The article is then concluded by stating that portability and interoperability are two key aspects when examining vendor lock-in and that it is almost impossible to completely eradicate it from occurring but with the right knowledge and with further research the problem can be more easily managed.

Discussion

The authors discuss that introducing uniform standards within the cloud computing domain is something that is not in the best interest of the CSPs. Our thesis work strengthens this claim since we also discovered in the literature studies that the major CSPs prefer proprietary solutions which allow them to develop with only their own preferences in mind. Rather the major CSPs consider a unified standard as an obstruction for their development as it creates a dependency on third party organizations which limits their potential of creating unique and competitive services. Opara-Martins et al. also suggest that the multiple standard bodies, forums and consortiums in the industry impose a risk of introducing multiple standards where no standardization approach is favored over the others. They suggest that in order to manage this problem it is necessary for the standard bodies, CSPs and customers to sit together in order to find a shared consensus of standards to be adopted. While this seems to be a good solution in theory, our findings indicate that the major CSPs are not interested in adopting a general standard for the above reasons and therefore we are not optimistic that these conversations will happen any time soon.

Opara-Martins et al. conclude their article by mentioning that they would like to further explore how different frameworks for cloud computing migration can address the challenge of vendor lock-in and potentially ease the development process. Our thesis contributes with knowledge in this area as we have discovered that third party frameworks in many cases solve the originating problem, vendor lock-in to the CSP. However, they also result in new problems such as dependencies on the frameworks being maintained and regularly updated by the third party organization. While these frameworks can ease the development process, we do not necessarily believe that they solve the vendor lock-in problem but rather move it from

the CSP to a third party organization, which is not necessarily better.

6.4.4 A Service-Oriented Framework for Developing Cross Cloud Migratable Software

The article by Guillen et al. aims to ease the development of applications that are to be deployed on multiple CSPs which align with the IKEA team's needs and motivation for adopting a multi-cloud strategy [38]. The authors discuss aspects such as abstractions that are an important part of our solutions, as well as their own framework for multi-cloud development. Although the framework is outdated, we consider many of their ideas and motivations to still be very relevant for our work.

Summary

The purpose of the article is to ease software development for multiple CSPs. Most CSPs offer tools that must be used or ease the development on their own cloud but this generally results in a vendor lock-in to the specific CSP. Previous work has tried to avoid the vendor lock-in by using abstraction layers or brokers that hide the differences between the CSPs. However, Guillén et al. claim that this results in another lock-in situation to the middleware which is not a satisfiable solution. The paper presents a development framework that should not result in a lock-in at low level. Developers develop as if applications are to be deployed on-premises and an additional deployment plan is created. The framework then utilizes a source code transformation engine to transform the source code into cloud artifacts. In order to validate their presented framework, it is applied to and tested in a real industrial project. They conclude that the framework has some clear strengths but also some weaknesses. While the development was improved in some aspects, the current number of supported CSPs greatly limited the development freedom. Also, they found that before the source code had been transformed by the transformation engine it had no connections to other services or databases which created new obstacles related to testing the code.

Discussion

Guillén et al. claim that their framework avoids coupling and vendor lock-in at a low level, i.e. in the source code. However, our findings from the thesis work suggest that other lock-ins such as mental lock-in and skill lock-in can be just as detrimental. Their framework instead affects the methodology of the development process since the new framework requires a certain work process to be functional. This is something that we consider to be a type of skill- and mental lock-in and we have found that the switching cost of these aspects are not necessarily lower than reworking the code to escape the low level lock-in.

Another aspect that we consider missing in their description of the framework is how they plan to maintain it. In our research, the main disadvantage we have found with third-party frameworks and transformation engines is that there is a substantial risk of being vendor locked out. If the framework is not maintained nor regularly updated, the customers will not be able to utilize new opportunities such as new services from the CSPs. Since Guillén et al. already found the limited support to be a problem during their experiment, we find

it difficult to see how a framework like this could be used in a larger company if it was not backed and maintained by a supporting organization. However, as we have previously discussed, the approaches and different methodologies these frameworks use are often useful and something that organizations should be aware of and consider as possible solutions for handling the problems themselves.

6.5 Future Work

During our thesis we have discovered several different ideas that we would have liked to further explore but due to the scope of the thesis and limited time we did not have the opportunity to do so. This section will discuss these ideas in order to inspire future research to continue exploring them further.

When we discussed the different challenges related to handling multiple CSPs we identified two different categories of challenges, technical and non-technical challenges. Due to the limited time and resources of the thesis we did not go further with exploring solutions for the non-technical challenges and solely focused on the technical ones in RQ2a. However, handling these challenges are something we consider important for an optimal adoption of a multi-cloud approach. We have discovered that several of these challenges potentially can impose a big threat and is something that an organization wanting to adopt a multi-cloud strategy has to be aware of. We therefore suggest that future research extends our RQ2a by taking these challenges into consideration when exploring possible solutions. For example, the question regarding how organizations accurately can choose the best CSP for a service from an economical perspective when their charging models differ and they come with different levels of free usage is one of many non-technical challenges that we consider essential to be explored and concluded for an optimal multi-cloud adoption.

As discussed in section 4.4, we consider SPLs and features models to be potential solutions for increasing the maintainability of variants in the context of multi-cloud but more research is needed. Due to SPLs being a research area on its own, we did not consider them to fit our scope and it would take too much time to properly explore their potential. We are therefore unsure exactly what they could bring to the maintainability of multi-cloud variants and what the cost of adopting them would be. We propose future research to extend our thesis work and investigate how they could be used within the area and if it is a solution that can be recommended.

A limitation of our hands-on experimentation was that we created our own application that was using similar services as those that the IKEA team is using but on a smaller scale. It was therefore difficult to determine how the development time of adopting a cloud agnostic solution scales with the complexity and size of the application. Future research could investigate how both the size and complexity of an application relates to the development time needed to adopt a cloud agnostic approach. The literature studies suggest that not all aspects of the development scale the same and that some aspects require the same amount of work no matter the size of the application whilst others can potentially scale unfavorably.

Our final suggestion for possible future work is to explore how our proposed solutions affect development and how easy they are to maintain over time. It is possible to argue that our hands-on experiments have a rather short perspective since they are mainly investigating how difficult our solutions are to implement and the immediate effects they have on the development process. While the literature considers the solutions to increase the maintainability, we consider the more long term perspective to not be fully explored and we encourage others, that have more time and resources than we had, to continue exploring what the long term effects of the solutions have on the maintainability and future extension of the code.

Chapter 7
Conclusions

We have identified the challenges associated with handling multiple CSPs and explored different solutions for how to address these challenges. In order to answer RQ1 we conducted a literature study and interviews. These revealed several challenges which could be categorized into two main categories, technical and non-technical challenges. The technical challenges were further categorized into four different subcategories; portability, interoperability, vendor lock-in and security aspects. In a similar fashion, the non-technical challenges were also categorized into four different subcategories; legal aspects, sustainability, economical and talent management. We identified portability as the major technical challenge when it comes to handling multiple CSPs. Regarding the non-technical challenges we recommend that they are further explored in order to optimize the use of multiple CSPs.

When investigating RQ2a we discovered that there exist a wide range of possible solutions to the technical challenges we had identified by exploring RQ1. We found that abstracting away the differences of the CSPs in many cases is the best practice. We also discovered that a cloud agnostic approach can greatly ease the handling of multiple CSPs but it comes with a cost of additional development time and limiting choice of services since only the least common denominator between the CSPs can be utilized. At the infrastructure level, we found that IaC tools, such as Terraform, result in a more uniform way of handling multiple CSPs.

When exploring RQ2b, we found variant segregation to be a suitable practice for managing the differences between CSPs. The use of variant segregation in combination with abstractions resulted in that we could implement a cloud agnostic main file that contained all the common non-CSP specific code while CSP specific parts could be handled as separate segregated variants that implemented the abstracts methods defined in the main file. Regarding RQ3, we consider our small application to be the proof of concepts that validates that the theoretical solutions identified can be practically implemented. We were also able to show that our solutions greatly reduced the time needed to implement support for an additional CSP in contrast to the time needed for implementing support for the second CSP.

References

[1] J. Abrahamsson Ring (CEO Inter IKEA Group), "IKEA grows again," 2021. https://about.ikea.com/en/about-us/year-in-review, [Accessed: 2022-05-05].

[2] Y. Zheng and Q. Wang, "Shadow of the great firewall: The impact of Google blockade on innovation in China," *Strategic Management Journal*, vol. 41, no. 12, pp. 2234–2260, 2020.

[3] A. Brown, "Facebook Lost About $65 Million During Hours-Long Outage." Forbes – Social Media – Editors' Pick. https://www.forbes.com/sites/abrambrown/2021/10/05/facebook-outage-lost-revenue/?sh=6a494491231a, [Accessed: 2022-04-21].

[4] W. Gawronski, "The Complete History of AWS Outages," 2022. https://awsmaniac.com/aws-outages/, [Accessed: 2022-04-29].

[5] "EDAN10 – Konfigurationshantering." https://kurser.lth.se/kursplaner/20_21%20eng/EDAN10.html, [Accessed: 2022-05-05].

[6] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, pp. 1–10, 2014.

[7] "Semi-structured interview." Wikipedia, 2021. https://en.wikipedia.org/wiki/Semi-structured_interview, [Accessed: 2022-05-02].

[8] "Pic-a-Daily Serverless Workshop." Codelabs, 2022. https://codelabs.developers.google.com/serverless-workshop, [Accessed: 2022-05-02].

[9] P. Mell, T. Grance, *et al.*, "The NIST definition of cloud computing," 2011.

[10] "Cloud computing." Wikipedia, 2022. https://en.wikipedia.org/wiki/Cloud_computing, [Accessed: 2022-05-02].

[11] J. Żmudziński, "Cloud Services Comparison 2022 – market share, main differences." Future Processing – Business Blog, 2022. https://www.future-processing.com/blog/cloud-services-comparison-market-share-main-differences/, [Accessed: 2022-05-02].

[12] J. Opara-Martins, R. Sahandi, and F. Tian, "Critical review of vendor lock-in and its impact on adoption of cloud computing," in *International Conference on Information Society (i-Society 2014)*, pp. 92–97, IEEE, 2014.

[13] S. Kolb and G. Wirtz, "Towards application portability in platform as a service," in *2014 IEEE 8th international symposium on service oriented system engineering*, pp. 218–229, IEEE, 2014.

[14] "FAQ - When should I choose a Cloud Native or Cloud Agnostic approach?." IKEA Digital Internal Information.

[15] R. Vettor *et al.*, "What is Cloud Native?," 2022. https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/definition, [Accessed: 2022-05-23].

[16] "Cloud Agnostic: What Does It Really Mean And Why Do You Need It?," 2021. https://www.cloudzero.com/blog/cloud-agnostic, [Accessed: 2022-05-23].

[17] W. A. Babich, *Software configuration management : coordination for team productivity*. Addison-Wesley, 1986.

[18] A. Mahler, "Variants: Keeping things together and telling them apart," in *Configuration Management*, pp. 73–97, 1995.

[19] R. Wang, "Adopting Cloud Computing: Seeing The Forest For The Trees." Forbes – Innovation, 2013. https://www.forbes.com/sites/oracle/2013/09/20/adopting-cloud-computing-seeing-the-forest-for-the-trees/?sh=1d55644838b9, [Accessed: 2022-04-22].

[20] J. Bozman and G. Chen, "Cloud computing: The need for portability and interoperability," *IDC Executive Insights*, 2010.

[21] R. Buyya, S. N. Srirama, G. Casale, *et al.*, "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–38, 2018.

[22] D. Petcu and A. V. Vasilakos, "Portability in clouds: approaches and research opportunities," *Scalable Computing: Practice and Experience*, vol. 15, no. 3, pp. 251–270, 2014.

[23] C. Ramalingam and P. Mohan, "Addressing semantics standards for cloud portability and interoperability in multi cloud environment," *Symmetry*, vol. 13, no. 2, p. 317, 2021.

[24] S. M. Barhate and M. Dhore, "Hybrid cloud: A solution to cloud interoperability," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 1242–1247, IEEE, 2018.

[25] M. Toivonen, "Cloud Provider Interoperability and Customer Lock-In," in *Proceedings of the seminar*, no. 58312107, pp. 14–19, 2013.

[26] D. Elliott, C. Otero, M. Ridley, and X. Merino, "A cloud-agnostic container orchestrator for improving interoperability," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pp. 958–961, IEEE, 2018.

[27] Y. Elkhatib, "Mapping {Cross-Cloud} Systems: Challenges and Opportunities," in *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.

[28] D. Petcu, "Portability and interoperability between clouds: challenges and case study," in *European conference on a service-based internet*, pp. 62–74, Springer, 2011.

[29] O. Tomarchio, D. Calcaterra, G. Di Modica, and P. Mazzaglia, "Torch: a tosca-based orchestrator of multi-cloud containerised applications," *Journal of Grid Computing*, vol. 19, no. 1, pp. 1–25, 2021.

[30] J. Opara-Martins, R. Sahandi, and F. Tian, "Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective," *Journal of Cloud Computing*, vol. 5, no. 1, pp. 1–18, 2016.

[31] F. Gonidis, A. J. Simons, I. Paraskakis, and D. Kourtesis, "Cloud application portability: an initial view," in *Proceedings of the 6th Balkan Conference in Informatics*, pp. 275–282, 2013.

[32] R. Ranjan, "The cloud interoperability challenge," *IEEE Cloud Computing*, vol. 1, no. 2, pp. 20–24, 2014.

[33] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, "TOSCA: portable automated deployment and management of cloud applications," in *Advanced Web Services*, pp. 527–549, Springer, 2014.

[34] K. Kaur, D. S. Sharma, and D. K. S. Kahlon, "Interoperability and portability approaches in inter-connected clouds: A review," *ACM Computing Surveys (CSUR)*, vol. 50, no. 4, pp. 1–40, 2017.

[35] D. Saxena, R. Gupta, and A. K. Singh, "A survey and comparative study on multi-cloud architectures: emerging issues and challenges for cloud federation," *arXiv preprint arXiv:2108.12831*, 2021.

[36] S. Dowell, A. Barreto, *et al.*, "Cloud to cloud interoperability," in *2011 6th International Conference on System of Systems Engineering*, pp. 258–263, IEEE, 2011.

[37] R. Yasrab and N. Gu, "Multi-cloud PaaS architecture (MCPA): a solution to cloud lock-in," in *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, pp. 473–477, IEEE, 2016.

[38] J. Guillén, J. Miranda, J. M. Murillo, and C. Canal, "A service-oriented framework for developing cross cloud migratable software," *Journal of Systems and Software*, vol. 86, no. 9, pp. 2294–2308, 2013.

[39] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *2010 24th IEEE international conference on advanced information networking and applications*, pp. 27–33, Ieee, 2010.

[40] S. Pearson and A. Benameur, "Privacy, security and trust issues arising from cloud computing," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pp. 693–702, IEEE, 2010.

[41] G. A. Lewis, "Role of standards in cloud-computing interoperability," in *2013 46th Hawaii international conference on system sciences*, pp. 1652–1661, IEEE, 2013.

[42] O. Spjuth, "Interoperability and scalability with microservices in science," 2016. `https://www.slideshare.net/ospjuth/interoperability-and-scalability-with-microservices-in-science`, [Accessed: 2022-05-09].

[43] N. Ferry, F. Chauvel, A. Rossini, B. Morin, and A. Solberg, "Managing multi-cloud systems with CloudMF," in *Proceedings of the Second Nordic Symposium on Cloud Computing & Internet Technologies*, pp. 38–45, 2013.

[44] T. Harmer, P. Wright, C. Cunningham, and R. Perrott, "Provider-independent use of the cloud," in *European Conference on Parallel Processing*, pp. 454–465, Springer, 2009.

[45] T. Mattila, "Comparing preconditions for cloud and on-premises development," *Cloud-Based Software Engineering*, p. 1.

[46] G. Hohpe, "Don't get locked up into avoiding lock-in." martinFowler.com, 2019. `https://martinfowler.com/articles/oss-lockin.html`, [Accessed: 2022-04-22].

[47] J. Hong, T. Dreibholz, J. A. Schenkel, and J. A. Hu, "An overview of multi-cloud computing," in *Workshops of the international conference on advanced information networking and applications*, pp. 1055–1068, Springer, 2019.

[48] J. Opara-Martins, "Taxonomy of cloud lock-in challenges," *Mobile Computing-Technology and Applications*, 2018.

[49] P. Lipton, "Escaping vendor lock-in with tosca, an emerging cloud standard for portability," *CA Labs Research*, vol. 49, 2012.

[50] R. Caldwell, "Pros and Cons of a Multi-Cloud Strategy." Centric Consulting Blogs, 2019. `https://centricconsulting.com/blog/pros-and-cons-of-a-multi-cloud-strategy/`, [Accessed: 2022-04-22].

[51] "Cloud Agnostic: What Does It Really Mean And Why Do You Need It?." CloudZero webpage, 2021. `https://www.cloudzero.com/blog/cloud-agnostic`, [Accessed: 2022-04-22].

[52] J. Jayalath, E. Chathumali, K. Kothalawala, and N. Kuruwitaarachchi, "Green cloud computing: A review on adoption of green-computing attributes and vendor specific implementations," in *2019 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, pp. 158–164, IEEE, 2019.

[53] I. Berry, "Top 10 countries with the most data centres," 2021. https://datacentremagazine.com/top10/top-10-countries-most-data-centres, [Accessed: 2022-05-06].

[54] E. Sayegh, "The Cloud Talent Drought Continues (And Is Even Larger Than You Thought)." Forbes – Cloud, 2020. https://www.forbes.com/sites/emilsayegh/2020/03/02/the-2020-cloud-talent-drought-is-even-larger-than-you-thought/?sh=7936862f58c0, [Accessed: 2022-05-06].

[55] J. Adam, "Understanding Cloud Agnostic Strategies – Why, Where, When and How?." Digital Transformation Blog, 2022. https://kruschecompany.com/cloud-agnostic-strategies/, [Accessed: 2022-05-09].

[56] T. Copado, "Cloud Agnostic vs Cloud Native: Developing a Hybrid Approach." Copado Articles, 2021. https://www.copado.com/devops-hub/blog/cloud-agnostic-vs-cloud-native-developing-a-hybrid-approach, [Accessed: 2022-05-09].

[57] A. Ranabahu, E. M. Maximilien, A. Sheth, and K. Thirunarayan, "Application portability in cloud computing: an abstraction-driven perspective," *IEEE Transactions on Services Computing*, vol. 8, no. 6, pp. 945–957, 2013.

[58] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, 2014.

[59] J. Watada, A. Roy, R. Kadikar, H. Pham, and B. Xu, "Emerging trends, techniques and open issues of containerization: a review," *IEEE Access*, vol. 7, pp. 152443–152472, 2019.

[60] "What are containers?." Google Cloud Topics. https://cloud.google.com/learn/what-are-containers, [Accessed: 2022-05-09].

[61] S. Ranjan, "Containers and cloud portability." Rackspace Technology Blog, 2018. https://docs.rackspace.com/blog/containers-and-cloud-portability/, [Accessed: 2022-05-09].

[62] "Containers in Cloud Computing: Enabling Portability, Agility and Automation." Cloud Native Wiki. https://www.aquasec.com/cloud-native-academy/docker-container/container-cloud-computing/, [Accessed: 2022-05-09].

[63] S. Rajan, "Containers and Cloud portability," 2019. https://www.linkedin.com/pulse/containers-cloud-portability-sriram-rajan, [Accessed: 2022-05-09].

[64] O. Tomarchio, D. Calcaterra, and G. D. Modica, "Cloud resource orchestration in the multi-cloud landscape: a systematic review of existing frameworks," *Journal of Cloud Computing*, vol. 9, no. 1, pp. 1–24, 2020.

[65] V. I. Munteanu, C. Şandru, and D. Petcu, "Multi-cloud resource management: cloud service interfacing," *Journal of Cloud Computing*, vol. 3, no. 1, pp. 1–23, 2014.

[66] K. Kritikos, P. Skrzypek, and F. Zahid, "Are cloud platforms ready for multi-cloud?," in *European Conference on Service-Oriented and Cloud Computing*, pp. 56–73, Springer, 2020.

[67] K. Morris, *Infrastructure as code*. O'Reilly Media, 2020.

[68] N. Deo, "3 Advantages and Challenges of Infrastructure as Code (IaC)." https://www.cloudbolt.io/blog/3-advantages-and-challenges-of-infrastructure-as-code-iac/, [Accessed: 2022-04-12].

[69] Y. Brikman, "Why we use terraform and not chef, puppet, ansible, saltstack, or cloudformation," *Retrieved April*, vol. 24, p. 2020, 2016.

[70] Y. Brikman, *Terraform: up & running: writing infrastructure as code*. O'Reilly Media, 2019.

[71] J. Thönes, "Microservices," *IEEE software*, vol. 32, no. 1, pp. 116–116, 2015.

[72] N. Dragoni, S. Giallorenzo, A. L. Lafuente, *et al.*, "Microservices: yesterday, today, and tomorrow," *Present and ulterior software engineering*, pp. 195–216, 2017.

[73] J. O. de Carvalho, F. Trinta, and D. Vieira, "PacificClouds: A Flexible MicroServices based Architecture for Interoperability in Multi-Cloud Environments," in *CLOSER*, pp. 448–455, 2018.

[74] C. Alliance, "Security guidance for critical areas of focus in cloud computing v3. 0," *Cloud Security Alliance*, vol. 15, pp. 1–176, 2011.

[75] M. Hosken, "It's Time to Develop a Cloud Exit Strategy," 2021. https://octo.vmware.com/its-time-to-develop-a-cloud-exit-strategy/, [Accessed: 2022-05-11].

[76] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 2013.

[77] D. Norman, "Application Monitoring Best Practices," 2021. https://www.prisma.io/blog/monitoring-best-practices-monitor5g08d0b, [Accessed: 2022-05-12].

[78] "Why Prisma Cloud?." https://www.paloaltonetworks.com/prisma/whyprisma, [Accessed: 2022-04-15].

[79] J. Walsh, "DevOps Security: Cloud Secrets Management, from Multi-Cloud to Cloud Agnostic Environments," 2021. https://www.conjur.org/blog/devops-security-cloud-secrets-management-from-multi-cloud-to-cloud-agnostic-environments/, [Accessed: 2022-05-12].

[80] "HashiCorp Vault." https://www.vaultproject.io/, [Accessed: 2022-04-22].

[81] "Avoid these common multi-cloud privilege access management mistakes," 2021. https://www.britive.com/blog/multi-cloud-privilege-access-management/?utm_source=rss&utm_medium=rss&utm_campaign=multi-cloud-privilege-access-management, [Accessed: 2022-03-21].

[82] M. Miller, "What is least privilege why do you need it?," 2021. https://www.beyondtrust.com/blog/entry/what-is-least-privilege, [Accessed: 2022-04-13].

[83] P. Clements and L. Northrop, *Software product lines*. Addison-Wesley Boston, 2002.

[84] D. Nešić, J. Krüger, Stănciulescu, and T. Berger, "Principles of feature modeling," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 62–73, 2019.

[85] K. Pohl, G. Böckle, and F. Van Der Linden, *Software product line engineering: foundations, principles, and techniques*, vol. 1. Springer, 2005.

[86] F. J. Van der Linden, K. Schmid, and E. Rommes, *Software product lines in action: the best industrial practice in product line engineering*. Springer Science & Business Media, 2007.

[87] E. Cavalcante, A. Almeida, T. Batista, N. Cacho, F. Lopes, F. C. Delicato, T. Sena, and P. F. Pires, "Exploiting software product lines to develop cloud computing applications," in *Proceedings of the 16th International Software Product Line Conference-Volume 2*, pp. 179–187, 2012.

[88] "Google vision api." `https://cloud.google.com/vision`, [Accessed: 2022-04-12].

[89] "What is cloud infrastructure?." RedHat, 2019. `https://www.redhat.com/en/topics/cloud-computing/what-is-cloud-infrastructure`, [Accessed: 2022-05-18].

[90] "About opencv." `https://opencv.org/about/`, [Accessed: 2022-04-15].

Appendices

Appendix A
Initial Set of Search Terms Used

In this appendix the initial sets of search terms used for the conducted literature studies are presented. Note that further keywords and search terms were later discovered by using the method presented in section 2.2.1.

A.1 Literature Study 1

- Multi-cloud
- Multi-cloud Challenges
- Cloud Computing Challenges
- Cloud Agnostic
- Vendor Lock-in
- Vendor Lock-in Cloud Computing

A.2 Literature Study 2

- Interoperability Cloud
- Portability Cloud
- Vendor Lock-in
- Security Multi-cloud
- Variant Handling Cloud Computing

- Variant Segregation
- Software Product Lines

Appendix B
Interview Template

The following appendix presents the interview questions we asked the developers of the IKEA team.

1. How are you working with CSPs on an everyday basis? (What's your role?)
2. We have understood that IKEA uses different CSPs and we are curious about how this works in practice. Can you describe this for us?
    - Does everyone in the team have knowledge about every available CSP or does a single person have responsibility for a specific CSP and is the general expert on his/her specific subject?
    - Are you comfortable working with all the CSPs IKEA uses?
    - Is the multi-cloud strategy affecting your daily work? If yes, how?
3. From your experience, which are the main challenges working with different/multiple CSPs?
    - In what situations do these problems occur in your everyday work?
4. During our literature study we have found some challenges and would like to hear your opinion regarding their severity and importance for IKEA.
    - Interoperability is a frequently used term in the literature regarding multi cloud. Have you heard about it before? If yes, can you try to describe it for us? We have defined interoperability as: *interoperability means the ability of two cloud systems to talk to another, i.e. to exchange messages and information in a way that both can understand.* According to the literature study, interoperability and more specifically the communications between different CSPs seem to be a major issue. Do you share this experience and if yes, when/how does it occur?

- Another term somewhat related to the previous one and discussed a lot in the literature is portability. Once again, are you familiar with this term? If yes, can you describe it for us? We have defined portability as: *portability means the ability to move data (files, documents, database tables, etc.) and executable software from one cloud system to another, and have that data and software usable and functional in the other system.* From your experience, is portability a real problem for you at IKEA?
- Security, according to some literature, there seems to be a lack of a standard between CSPs which makes it difficult to establish a common security framework across multiple CSPs. How do you work with these security issues?
  - First of all, in your work are you actively thinking about security issues related to the cloud?
  - Is security issues different for the CSPs?
  - Is it possible to make a "one size fits all" solution for security?
- Have you found it difficult to find talent for specific CSPs or is everyone able to work with all CSPs?

5. Are you satisfied with the current solutions present at IKEA for these problems?
   - Which parts are working well?
   - Do you think there is room for improvement related to multi-cloud?
6. We understand that IKEA strives to be completely cloud agnostic. What do you see as the main challenges as of today for achieving this goal? And is it even possible?

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2020-06-10

EXAMENSARBETE Handling Multiple Cloud Service Providers – Common Challenges and Best Practices
STUDENTER Pontus Jaensson, Oscar Wiklund
HANDLEDARE Lars Bendix (LTH), Habib Un Nabi Hillol (IKEA)
EXAMINATOR Per Andersson (LTH)

Hur användning av flera molntjänstleverantörer bör hanteras

POPULÄRVETENSKAPLIG SAMMANFATTNING **Pontus Jaensson, Oscar Wiklund**

Globala organisationer måste använda sig av fler än en molntjänstleverantör för att nå ut till alla världens marknader på ett effektivt sätt. Detta medför dock flera utmaningar som måste hanteras. Vårt arbete kartlägger de mest frekvent förekommande utmaningarna samt bästa praxis för att adressera dessa.

Onlineapplikationer har höga förväntningar från dagens kunder när det kommer till både tillgänglighet och prestanda. För att hantera detta har globala organisationer allt mer övergått från att själva ansvara för exekveringsmiljön till att outsourca denna del till externa molntjänstleverantörer. Detta göra att applikationer kan köras nära kundens fysiska plats utan att företagen själva behöver investera i nödvändig infrastruktur världen över. Problemet är att dessa molntjänstleverantörer inte finns tillgängliga i alla marknader som företag är verksamma i och därför tvingas organisationer i vissa fall använda sig av flera molntjänstleverantörer.

Då användandet av flera olika molntjänstleverantörer är ett relativt nytt område så har vårt arbete undersökt vilka utmaningar som detta medför samt bästa praxis för att hantera dessa. Vi har genom en litteraturstudie och intervjuer kartlagt vilka de främsta utmaningarna är och kom fram till att det går att kategorisera utmaningarna som antingen tekniska eller icke-tekniska. Den främsta tekniska utmaningen visade sig vara portabiliteten av applikationen, förmågan att kunna flytta applikationen mellan molntjänstleverantörer. Resterande delen av vårt arbete syftade främst till att hitta lämpliga sätt att adressera detta problem och därmed öka portabiliteten.

Genom en ytterligare litteraturstudie och egna experiment kom vi fram till att användandet av flera olika molntjänstleverantörer kan ses som olika varianter av samma applikation. För att utveckla och underhålla dessa varianter på ett effektivt sätt kom vi bland annat fram till att abstraktioner och isolering av de molntjänstspecifika delarna är lämpliga tillvägagångssätt. Isoleringen gör att vi får en gemensam kodfil som är helt oberoende av vilken molntjänstleverantör som är tänkt att användas. Denna fil kompletteras sedan med en fil som innehåller de molntjänstspecifika delarna som behövs för att kunna köra applikationen på den tilltänkta leverantörens infrastruktur. Utöver detta har vi även identifierat andra tillvägagångssätt som adresserar de resterande tekniska utmaningarna som identifierades.