INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Pragmatic Approaches from On-Premise to Cloud

PEDRO MIGUEL OLIVEIRA QUINTÃ
Outubro de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Pragmatic Approaches from On-Premise to Cloud

Pedro Miguel Oliveira Quintã

Dissertation to obtain the master's degree in Informatics Engineering, Area of Expertise in Software Engineering

Advisor: Constantino Martins

Supervisor: Miguel Veiga

Jury:

President: Alexandre Gouveia

Jury member: Alexandre Bragança

Porto, September 2022

Dedicatory

“I dedicate this work to all my family members and work colleagues who have always supported me in my academic and professional journey. Without them, nothing would be possible.”

Resumo

Na última década, cada vez mais empresas procuram utilizar cloud pública como meio de implantação dos seus serviços. Essa procura deve-se em grande parte à possibilidade de custos de infraestrutura mais baixos, melhor usabilidade para os utilizadores e um desejo crescente de desenvolver e evoluir para agradar o utilizador e, desta forma, rentabilizar o seu negócio.

A Cloud Pública é um modelo de implantação oferecido por um provedor de serviços que administra e gera toda a infraestrutura. Neste tipo de *cloud*, a customização é limitada porque é um serviço compartilhado por diversas empresas, sendo idealizado consoante requisitos da maioria dos seus clientes. Apresenta grandes vantagens relativamente a uma *cloud on-premise* como, por exemplo, o cliente não ter de se preocupar com a manutenção da infraestrutura, focando-se apenas nos serviços e no código aplicacional, apresentando um produto melhor ao seu público alvo. Estes pontos positivos levam a uma grande procura por este tipo de cloud mas traz algum trabalho de migração que por vezes pode apresentar-se como pesado ou problemático.

Os principais problemas quando se analisa a migração de serviços entre cloud privada e cloud pública abrangem diversos pontos como a escolha do modelo de implantação, a escolha do modelo de serviço, o entender se um serviço pode ser migrado diretamente ou se necessita de alterações na sua base, entre outros. Todo este conjunto de problemas leva a um receio dos clientes sobre este tipo de migração, havendo a falta de um modelo claro que facilmente resolva e delineie todo este processo.

Este trabalho tem como objetivo estudar e desenvolver uma metodologia para migração de componentes entre cloud privada e cloud pública, descrevendo todas as etapas desde o planeamento até a manutenção, incluindo claro a migração, de forma a ajudar qualquer empresa na migração dos seus componentes, tirando todo o receio neste tipo de mudança.

Palavras-chave: Cloud Pública, Cloud Privada, Migração, Plano de Migração, SaaS, IaaS, PaaS

Abstract

In the last decade, more and more companies are looking to use public cloud as a means of deploying their services. This demand is largely due to the possibility of lower infrastructure costs, better usability for users and a growing desire to develop and evolve to please the user and, in this way, make their business profitable.

The Public Cloud is a deployment model offered by a service provider that manages and generates the entire infrastructure. In this type of cloud, customization is limited because it is a service shared by several companies, being idealized according to the requirements of most of its customers. It has great advantages over an on-premise cloud, such as the customer not having to worry about infrastructure maintenance, focusing only on services and application code, presenting a better product to its target audience. These positive points lead to a great demand for this type of cloud but it brings some migration work that can sometimes be presented as heavy or problematic.

The main problems when analysing the migration of services between private cloud and public cloud cover several points such as the choice of the deployment model, the choice of the service model, understanding if a service can be migrated directly or if it needs changes in its base, among others. All this set of problems leads to a fear of customers about this type of migration, with the lack of a clear model that easily solves and outlines this entire process.

This work aims to study and develop a methodology for migrating components between private cloud and public cloud, describing all the steps from planning to maintenance, including of course the migration, to help any company in the migration of its components, taking all fear in this type of change.

Keywords: Migration Plan, Migration, Public Cloud, Private Cloud, SaaS, IaaS, PaaS

Acknowledgments

This thesis would not be possible without the support and backing of many who were directly or indirectly involved in this process.

Special thanks to all my family and friends for their support in this academic journey that culminated in this thesis. Without them, it would not have been possible.

A note of thanks to Instituto Superior de Engenharia do Porto and to all the teachers, especially Professor Constantino Martins, for all the academic path and for the help and teaching in the development of this thesis.

Last and not least, a thank you to all my colleagues at Blip who have always supported me during my master's degree and who on a daily basis have directly or indirectly supported me in this project. More than colleagues, they became friends and I could not fail to thank them for their support. A very special thanks to Miguel Veiga that since the beginning of my professional path taught me and helped me with all the problems and, without any hesitation, accepted to be my supervisor in this thesis and helped me to guide the project in the right direction giving me the necessary knowledge bases.

Table of Contents

Figure List

Table List

Acronyms e Symbols

Acronyms' List

SMEs	Small and Medium Enterprises
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
SBS	Springboot Service

1 Introduction

This chapter provides an introduction about the topic that will be covered in this thesis, taking a closer look about Public Cloud and the difference between this cloud type and Private Cloud.

Besides that, this chapter will also address the problem, the objectives and a briefly explanation of the used approach and the expected results.

1.1 Context

Application growth leads to the need for infrastructure that allows easy deployment of services and, at the same time, allows to profit from it and to have a proximity to customers for a quick response.

For many years, the vast majority of companies had their own servers, in which they had to make a high initial investment and which led to the need to hire professionals to maintain them well and ready to run their applications. This model led to a high initial investment and a constant cost of maintenance, replacement and improvements that sometimes did not generate the greatest profit.

As the years progressed, technology evolved and server supply companies appeared, which presented simpler, faster and lower cost solutions that enabled the deployment of services for various companies. This model allowed a reduction of investment for many organisations, as they no longer needed to buy servers, maintain them or improve them over the years, but only had to pay a contractual fee for the use of these providers' servers.

Blip is one of these cases of a company that began by owning its own servers and in recent years has been exploring public cloud services to improve its costs, performance and proximity to the customer. Blip is a tech company founded in 2009 in Oporto, with the objective of developing quality software in a casual and familiar environment. It’s one of the top software engineering companies in Portugal and plays an essential part in a bigger group called Flutter Entertainment, which includes betting sites such as Betfair, Paddypower, Fanduel, Sportsbet, among others.

Blip is responsible for the majority of the components that support all the betting for sites like Betfair or Fanduel. As mentioned above, all the Europe Infrastructure is mainly on-premise cloud, supported by servers maintained by the group in two datacenters and thousands of components deployed in tens to hundreds of hypervisors. The entire infrastructure is managed using OpenStack to maintain and manage all the Virtual Machines. For infrastructure outside the European Union, a mix between Private and the Public Cloud is used.

Blip supports several components that are at the heart of many of the biggest betting sites in the world. Its infrastructure is mostly on-premise, which leads to a higher maintenance cost, not only due to the internal framework that supports deployment but also due to upgrades, hypervisor changes or changes in datacenters.

1.1.1 Public Cloud

Public Cloud is a Deployment Model known for representing a service offered by a cloud provider to the general public. It is operated, managed and owned by a private company that make the resources available to multiple customers. In this type of cloud, the customization is limited since the infrastructure is shared amongst many customers and the service is designed with an idea of the requirements of most customers.

As main positive points, they have the advantage that the customer will not have to maintain the infrastructure, but only the code of its components and, depending on the chosen service model, configurations and/or tooling of the service, the cost advantage may vary in most cases.

1.2 Problem

Although Public Cloud is increasingly being talked about and there has been a notable growth in the adoption of this cloud deployment model, there is still no clear migration plan that allows any company to know the steps it must take from preparation to migration, through later maintenance and stabilization of the component in the Public Cloud.

The main problems related to deployment or migration of components are:

- Choose the deployment model (Public, Private, Hybrid, Community)
- Choose the service model (IaaS, PaaS, SaaS)
- Find out if there is a service that can fully replace an already existing service maintained by the company (related to SaaS)
- Know if it is necessary to make any changes to the component (Refactor) or if it is enough to move it to the new cloud (Forklifting)
- Know the differences between infrastructures for Public and Private Cloud

To make this process easier, it is necessary to have a model that not only assesses whether a component should be migrated but also identifies the steps that must be carried out from system specifications definition to component deployment and maintenance.

Removing this burden of long migrations and not knowing the steps to follow, you can not only save time and money, but also give developers greater freedom to make improvements and new features in the system.

1.3 Objectives

The main objective of this dissertation is to explore several existing models of migration between on-premise and public cloud, developing a more complete model that manages to have all the good points of existing models. With this model, the objective is to facilitate the migration process and also, in some cases, to adapt or improve the components.

It is intended to test the model designed, applying it in the migration of services and later evaluate possible improvements of this model.

1.4 Approach Taken

The main approach, comprises a study that will be carried out for several migration models existing in the market, based on multiple scientific documents. Afterwards, a comparison of the technical characteristics of each model will be performed.

The construction of the main model will be based on the results of the previous analysis, with a view to obtain the most optimal solution for all the available information, considering all the essential aspects and phases for the migration of services. The objective will be to select the positive points of all the models and, together with the requests from stakeholders, try to create a model that contains steps from the preparation to the post migration of the services.

After the initial design, each of the defined steps will be detailed and the model will be applied in the migration of a service, taking metrics and migration time values, performance and evaluation of the model itself.

There will be also some complementary tests to support this model, particularly about migrating components from on-premise cloud to public cloud.

1.5 Research Methodology

For the research, it will be conducted an analysis of scientific documents obtained from the ACM digital library (*ACM Digital Library*, n.d.) will be carried out, using the following keywords: cloud, cloud migration, public cloud, private cloud, on-premise cloud, AWS, IaaS. The Technical information will also be collected from internal company documents for the purpose of comparison with scientific documents obtained in this research.

For some specific cases, notes and documents will also be taken from companies' websites and/or videos of them in lectures and technological events.

1.6 Expected Results

This document aims to develop a complete migration model from on-premise to public cloud, with all the steps from planning to migration, not excluding identification of requirements with stakeholders and maintenance and monitoring of components after migration.

To do so, it will use the case study research method, by identifying some existing models and evaluating the value of each one to make it easier to gather ideas and values from existing models. A clear and detailed model is expected, which will be complemented with the development of some sub models for specific cases as for several identical technologies or components.

1.7 Document Structure

This document will present an analysis of the theory regarding migration between clouds, addressing existing methodologies in the market and designing the entire migration process planned in the objectives section.

Initially, a state of the art will be presented along with technical information about the different cloud service and deployment models existent, also addressing ideas from various existing methodologies.

Then a value analysis will be presented, comparing the existing methodologies studied previously in the state of the art with the idea that will be developed afterwards. This analysis will make it possible to depict which are the best methodologies to have as a reference for the development of the project.

Finally, an experimentation and evaluation chapter is elaborated, exploring models of evaluation of results obtained from the application of the methodology and migration plan.

2 State of Art

This chapter provides a quick overview about cloud computing nowadays, giving some key points on public and private cloud platforms, followed by a study on different approaches and frameworks for migration between the two types of cloud platforms. More information regarding Cloud, can be found in Appendix A – Cloud Study as supporting material for the development of this thesis.

Much of the technical information in this chapter will address content from books and scientific documents perspectives like Architecting The Cloud – Design Decisions For Cloud Computing Service Models (Michael Kavis, 2014), The Enterprise Cloud – Best Practices for Transforming Legacy IT (James Bond, 2015) and Cloud Native Infrastructure (Justin Garrison and Kris Nova, 2018).

2.1 Cloud Computing State

Before talking in depth about Cloud Computing, first it is necessary to define what the term Cloud Computing means, as well as describe the history of computing over the last 60 years or so.

Cloud computing is the technology of delivering hosted services over the internet (Balobaid & Debnath, 2020), which has gradually evolved over the years dating back to the first computers. The National Institute for Standards and Technology (NIST) defines the cloud computing as "*a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*" (James Bond, 2015).

Although there is this definition, many consider the cloud just another term for the Internet (James Bond, 2015), which may or may not be correct because it can include the use of computing resource over the Internet.

2.1.1 Cloud Characteristics and IT Goals

Following the National Institute for Standards and Technology (NIST) Cloud Computing definition, there are five essential characteristics that should be considered when talking about cloud technology (Mell & Grance, 2011):

Characteristics	Definition
On-demand self-service	Ability to expand services or resources as needed without requiring human interaction from the service provider (Mell & Grance, n.d.). This is a very important characteristic because gives the opportunity to scale up or scale down when needed, in times of greater or lesser need of resources.
Broad network access	Availability over the internet and accessed by thin or thick client platforms (computers, laptops, mobile devices, between others), over a private network communication circuit or over the Internet, depending on the cloud deployment model (explored later in this (James Bond, 2015).
Resource pooling	Users share all resources within a specific cloud deployment, and these are distributed using a multi-tenant model, with different physical and virtual resources. The location of the resources might not even be known by the user, depending of the deployment model. (Mell & Grance, 2011)
Rapid elasticity	Ability to scale rapidly outward and inward with demand, giving the impression of unlimited resource availability.
Measured service	Cloud systems automatically control and optimize the use of resources like memory, storage and network, leveraging a metering capability according to the use type (pay-per-use or charge-per-use basis) (Mell & Grance, n.d.).

Table 1 – Cloud Characteristics

All these features must be taken into account when deciding on the model and use of the cloud. When the components and services that a company wants to deploy on the cloud are critical, there are some additional requests and specifications that could be requested (James Bond, 2015):

Characteristics	Definition
Real-time statistics and monitoring	Ability to access real-time monitoring of service status, resource utilization dashboards, and a service-level agreement (SLA) reporting scorecard
Self-service management	Ability to manage accounts and application settings after the application is moved to the cloud
Role-based security for multilevel administration	Definition of user roles and permissions, allowing the company to manage the visibility of services between users so there could be multiple organizations and tenants.
24 hours Support	Fully support for all the services and resources hosted in the cloud
Little or no capital expenses	Since in a public cloud, the initial cost of the equipment is supported by the cloud provider, customers can better utilize their money and not having to pay for equipment and upgrades over multiple years.
No long-term commitments and the ability to easily scale services up or down as needed or consumed	Customers often desire little or no minimums or term commitments which, depending on the terms of the contract, may or may not be agreed with the provider. Often, to obtain the desired conditions, the result is a higher price per service over a period of time.

Table 2- IT Cloud Requests

2.1.2 Cloud Computing Models

Before exploring cloud computing models, it is important to have an overall view of the cloud model for an easier understanding about the main topics to be addressed (James Bond, 2015):

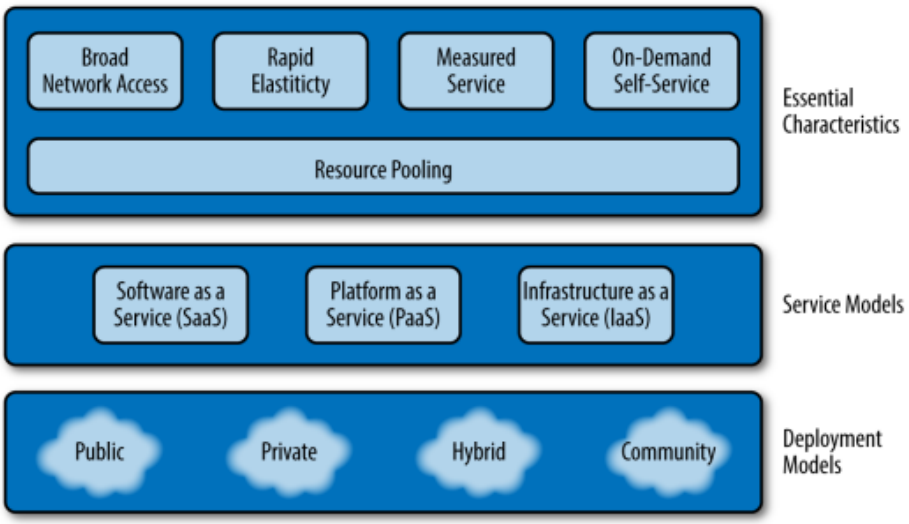


Figure 1 - NIST Cloud Computing Model (Mell & Grance, 2011)

As shown above, the Resource Pooling is an important part of the Essential Characteristics and influences the rest.

The following two models (Service and Deployment) will be explored in the topics below and will have a relevant weight in the research made in this document.

2.1.2.1 Service Models

When deploying a service to the cloud, there are three types of models that can be used and that should be analysed, depending on the component in question and the requirements defined by the customer:

- **Software as a Service** (*SaaS*) - Applications are hosted and managed by the provider in the cloud with the consumer accessing the application services from various client devices through either thin client interfaces or a program interface. The provider can sometimes provide a control panel so the consumer can have some control over certain aspects of the application or configurations but the consumer does not manage any underlying cloud infrastructure like network, servers, storage, among others (Mell & Grance, 2011).
- **Platform as a Service** (*PaaS*) - The provider manages all underlying virtual machines (VMs), networking, storage, operating system, and core applications. The consumer can deploy applications using programming languages, libraries, services and tools supported by the provider(Mell & Grance, 2011). As with SaaS, the customer will not manage anything related to networks, processing or storage and will not be responsible for the operating system too. The customer's sole responsibility will be the service/application and its settings.
- **Infrastructure as a Service** (*IaaS*) - In contrast with *PaaS*, in this case the consumer can deploy the operating system and the applications he/she wants. The cloud provider handles the management of the underlying infrastructure including networking,

storage systems, and virtualization/hypervisors, while the consumer manages the OS, applications, and data. Sometimes the provider can supply OS templates as a help to the customer to make their work easier.

To make easier this comparison between models, the following table will summarize the points each one of them present to the customer (Judith Hurwitz et al., 2012):

Model	Provider Managed	Customer Managed	Pricing
Software as a Service (SaaS)	• Underlying cloud infrastructure (network, storage, servers, …) • Operating System • Application	• Control over certain aspects of the application or configurations	• Often based on a per-seat user license with potential up-charges for additional storage or application features
Platform as a Service (PaaS)	• Underlying cloud infrastructure (network, storage, servers, …) • Operating system • Core Applications /Resources	• Application	• Combination of the compute cost, the licensing costs of a database application, and the per-user fees, as well as additional storage or application features
Infrastructure as a Service (IaaS)	• Underlying cloud infrastructure (network, storage, servers, …)	• Operating System • Application	• Based on processor, memory, storage, and network resources, with established limits or the ability to automatically scale up

Table 3 – Service Models Summary

2.1.2.2 Deployment Models

Sometimes even more important than choosing the service model, choosing the right deployment model can affect an entire company and all of its components as opposed to the service model that can vary between services. Before moving on to this choice, it is necessary to consider the four existing models (Judith Hurwitz et al., 2012):

- **Public Cloud** – Cloud Service offered by a cloud provider to the general public. It is operated, managed and owned by a private company that make the resources available to multiple customers. In this type of cloud, the customization is limited because it is shared across many customers and the service is designed with an idea of the requirements of most customers.
    - **Virtual Private Cloud (VPC)** – Variation of public cloud, represented by a compartment from a public cloud and dedicated to a specific customer. This offers the advantages of using public cloud (like the pricing) and, in addition, some customization and VMs, Storage and Networking like a Private Cloud.

- **Private Cloud** - The cloud infrastructure is provisioned for exclusive use by a single organization, and it can be hosted on premises or at a third-party datacenter (Mell & Grance, n.d.). Typically, it is more customizable than other cloud forms because is created and used by a single customer/organization.
- **Community Cloud** – Cloud Infrastructure designed and provisioned for exclusive use of a community of consumers that have shared concerns and requirements (Mell & Grance, n.d.). It may be owned by one or more of the organizations in the community, or by a third-party company. Sometimes this term is used in the marketing to explain a service to a customer, although the cloud might technically be a VPC, a hybrid or a private cloud(James Bond, 2015).
- **Hybrid Cloud** – Combination of two or more of the previously deployment models that are bounded together by standardized or proprietary technology (cloud management system or cloud broker system) that enables data and application portability.

Figure 2 represents the relationship between the different models and the differences between them. If the customers use and connects to one or more types of cloud providers, a hybrid cloud is formed.

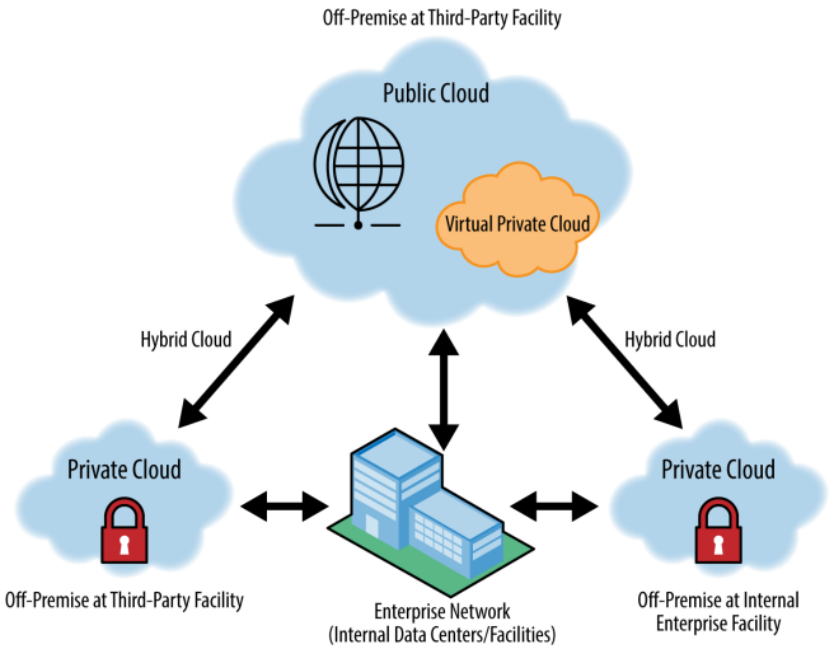


Figure 2 - Cloud Development Models (James Bond, 2015)

2.2 Public and Private Cloud Use from SMEs to Big Companies

After getting an idea of the concepts and importance of the cloud nowadays, it is also important to understand the impact it has on companies, from the smallest to the largest ones such as Google, Facebook or Spotify.

All the choices have a cost for the company and, therefore, all of them have to be made taking into account valid bases and that allow the company to generate income in the future. These impacts are often compared and illustrated by Conway's Law which states "If the architecture of the system and the architecture of the organisation are at odds, the architecture of the organisation wins" (Justin Garrison & Kris Nova, 2018). If the company wants the change from on-premise model to elastic or hybrid model to be the most clear and smooth as possible, there must be an overall organization change and planning so that all changes and effects that this shift can affect the product are considered. The Conway's Law can be represented in the Figure 3 below.

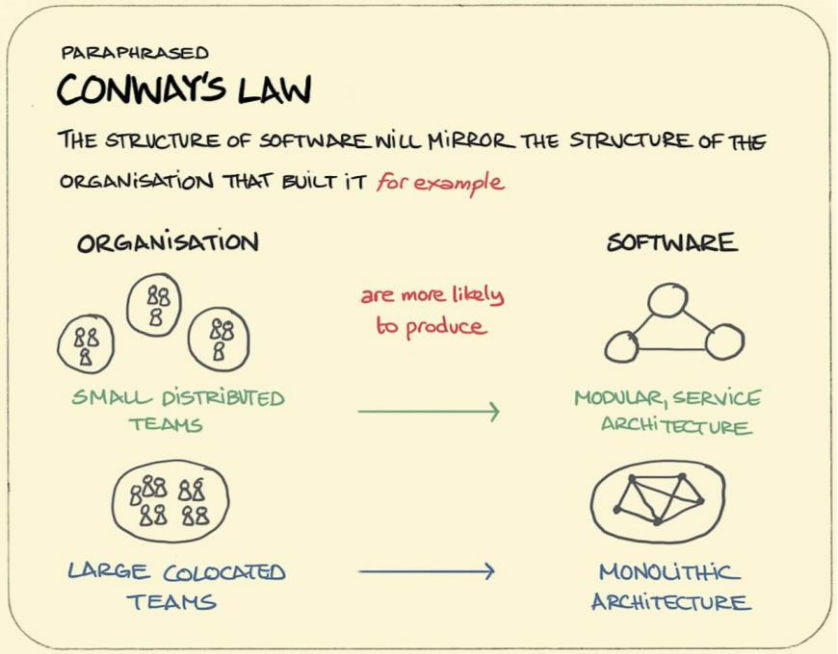


Figure 3 - Conway's Law (Jono Hey, n.d.)

For these reasons, all Cloud migration work presents various risks and requires extra care so that the development and deployment is done with good foundations and with a vision for the future for both the service and the company.

More information about these topics and all the investigation made during this project can be found in the Appendix A – Cloud Study.

2.3 Documented Cloud Migration Techniques

Over the last few years, more and more companies have shown an interest in migrating components to the public cloud, as these services presented great advantages for their business. But not everything is positive and migration often became a big headache.

To combat this migration problem, several researchers and companies studied migration methodologies and techniques, some of which will be mentioned below.

All the documents analysed were collected from the ACM digital library (*ACM Digital Library*, n.d.), using keywords such as "cloud migration", "cloud metamodel", "private to public cloud", "private cloud" and "public cloud".

2.3.1 Framework for SMEs – Migration Plan

This framework was developed and studied by two engineering's from the United States that noticed in the market a lack of knowledge about migration methods between Clouds, which caused a fear in exploring this alternative mainly on the part of small and medium companies (SMEs). These companies considered cloud migration to be a costly solution as compared to their on-premises services and fear a lack of security (Balobaid & Debnath, 2020). Therefore, they proposed a framework that would illustrate the entire process of cloud migration, from initial consideration to the post-migration steps.

Before any work, they explored the different Service Models (SaaS, IaaS, IaaS) and Deployment Models (Private Cloud, Public Cloud, Community Cloud and Hybrid Cloud) to understand and document the options each company has when thinking about migrating their services. A company may choose one deployment model for their services or infrastructure, or a combination of more than one model depending on the services they are trying to migrate (Balobaid & Debnath, 2020).

They also explored and documented some challenges that were also due to the lack of knowledge from SMEs. As documented before in this state of art, the interoperability between systems, the mapping of services between systems, the data privacy and security and also the Cost were the main challenges identified in the document.

2.3.1.1 Proposed Framework

"Though there have already been many frameworks available for the cloud adoption/migration, they are not comprehensive and detailed, and fail to cover the post migration processes especially for SMEs" (Balobaid & Debnath, 2020).

The proposed framework had 3 main stages: Pre-Migration, Migration and Post-Migration. Each one had some minor stages identified (Balobaid & Debnath, 2020):

1. **Pre-Migration** - focus on planning, preparation, and determining how existing IT infrastructure will fit into the cloud

a. Service Discovery - list all the running services in the infrastructure that the enterprise is willing to migrate to the cloud
b. Feasibility/assessment – examine the service offered by different providers on which the required services could be migrated, and to select the best of them.
    i. Service Provider Selection – study of different cloud providers and their offer. This phase can be divided in several small steps depending on some parameters like Credibility, Offering, Availability, Costing, between others.
    ii. Service Mapping – this stage is mainly about identifying the service and deployment models, defining the migration strategy and addressing some possible migration problems and difficulties.

2. Migration

a. Data/Services Backup – data backup from the services to be migrated and then stopping service on-premises for Data Integrity.
b. Deployment – this stage is composed by several steps that will rollout the service to the Cloud. It includes networking setup, definition of environment and service providing.
c. Restore/Replicating Services - restoring backed up services from on-premises to provisioned cloud services

3. Post-Migration

a. Post-Migration Steps – tuning the deployed services to have the best performance and overcome any failure
    i. Security Implementation
    ii. Backup Scheduling
    iii. Enable Monitoring
    iv. Validation/Testing
b. Go Live – configuration of network routers to serve the user traffic
c. Manage and Operate – long and continuous process of managing and operating the systems, monitoring the system and optimizing and upgrading.

After the framework development, that was a clear conclusion that one important part that needed to be explored more would be the post-migration because it needs a better plan depending on the system that may include monitoring, optimization and specific alerts.

2.3.2 Cloud Migration Metamodel

This metamodel was developed by three Computer Science Researchers Pooja Parnami, Aman Jain and Navneet Sharma. It is more focused in creating an abstract view of the generic migration process, presenting some steps and representing all the stages using UML notations.

Before defining the metamodel, there was some research and documentation, arriving to a concept defined in three phases:

1. Pre-Migration – initial migration tasks, including organization policies, feasibility study, definition of technical and non-technical requirements between other processes.
2. Design – Definition of the migration plan, Choice of Cloud Provider and identification of changes and design validations needed for the components that would me migrated.
3. Migration – Design and Deployment of Cloud Services, Validation, Monitoring and Optimization.

Each phase gave rise to a UML model that represents the various tasks and processes that compose it.

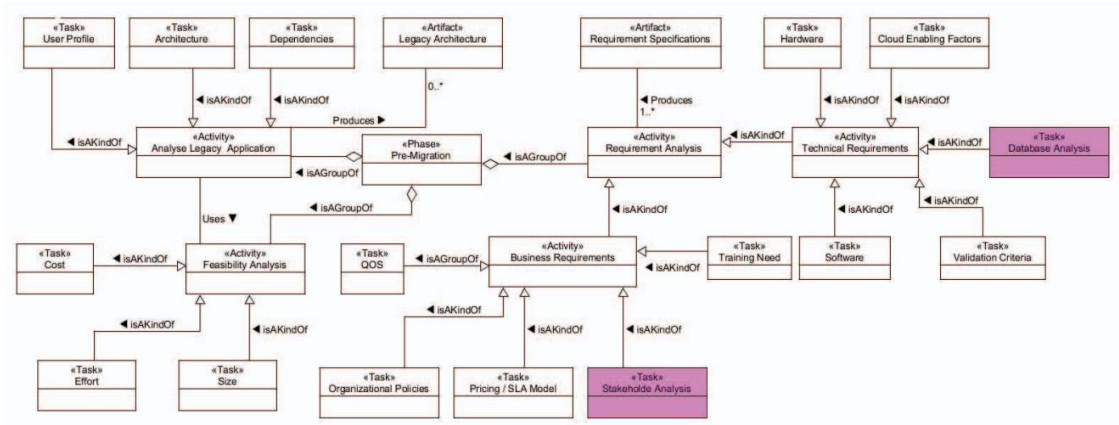


Figure 4 – Cloud Migration Metamodel – Pre-Migration Phase (Pamami et al., 2019)

The Pre-Migration Phase, as previously mentioned, is composed of tasks related to problem analysis, requirements gathering, architecture definition and feasibility analysis.

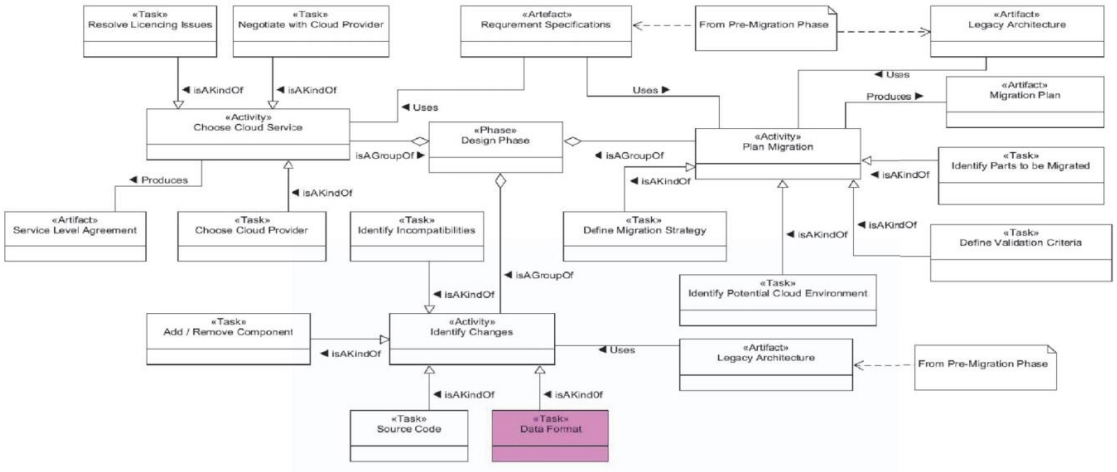


Figure 5 - Cloud Migration Metamodel – Design Phase (Pamami et al., 2019)

The Design Phase includes tasks such as defining the migration plan, choosing the cloud provider, identifying possible changes and incompatibilities, among other tasks necessary before carrying out the migration itself.

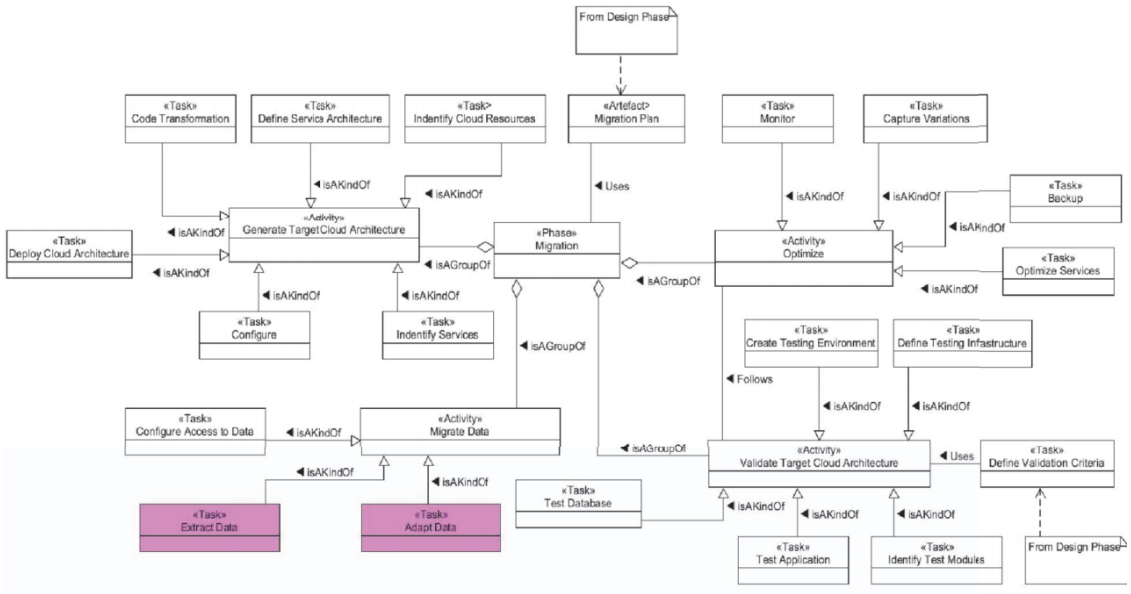


Figure 6 - Cloud Migration Metamodel – Migration Phase (Pamami et al., 2019)

The last phase will be the migration itself and includes the migration, optimization tasks and validation of applications.

2.3.3 Cloud Adoption Toolkit

This Case study was done to demonstrate the potential benefits and risks associated with the migration of a specific IT system (in this case, a IT system in the oil and gas industry) from a in-premise server to an Amazon EC2 Server (Khajeh-Hosseini et al., 2010). The paper was mainly focused in IaaS because is "arguably the most accessible to enterprise as they could potentially migrate their systems to the cloud without having to change their applications" (Khajeh-Hosseini et al., 2010). It starts by talking about the proposed migration and the project itself, talking later about the approach methodology.

One of the main topics in the analysis was the security risks of using Public Cloud to a company that needs their oil and gas data secure according to the laws for UK-Based Organizations.

An analysis of the benefits and risks of this migration was carried out and a decision-making support model was designed based on the analysis of cost and technology and a great support in the ideas and opinions of the stakeholders. The tool was called *Cloud Adoption Toolkit* and can be represented by the diagram in Figure 7.

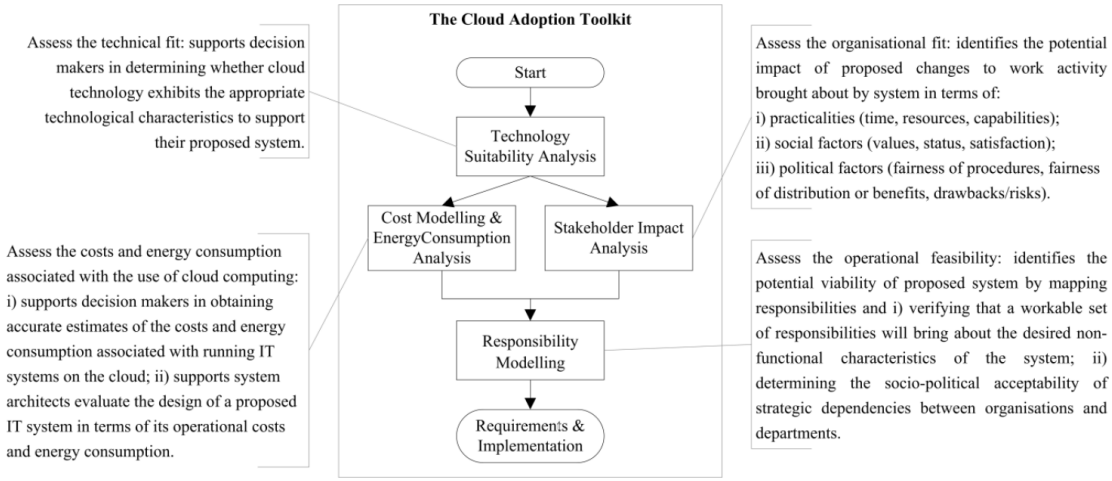


Figure 7 – Cloud Adoption Toolkit (Khajeh-Hosseini et al., 2010)

This Toolkit starts by analysing the technical specifications and the cloud characteristics, trying to understand if this solution answers the problem they intend to solve. After this stage, there will be two parallel analyses: Cost/Energy Analysis and Stakeholder Impact Analysis. The first one will mainly assess the cost and energy consumption, trying to obtain accurate estimates of the cost the company will be have with the migration. The Stakeholder Impact Analysis is one important and interesting stage of this tool because will gather a group of stakeholders and make a group of questions to every one of them. With the results, it is possible to create some statistics and analyse what is the expected impact for the stakeholders, taking some actions to minimize their discomforts.

2.3.4 LivCloud

LivCloud is a live cloud migration approach that does not need the provider's agreement to be applied. It is designed based on live cloud migration criteria to achieve effective live migration of VMs public cloud infrastructure (IaaS) without service interruption (I. E. A. Mansour et al., 2017).

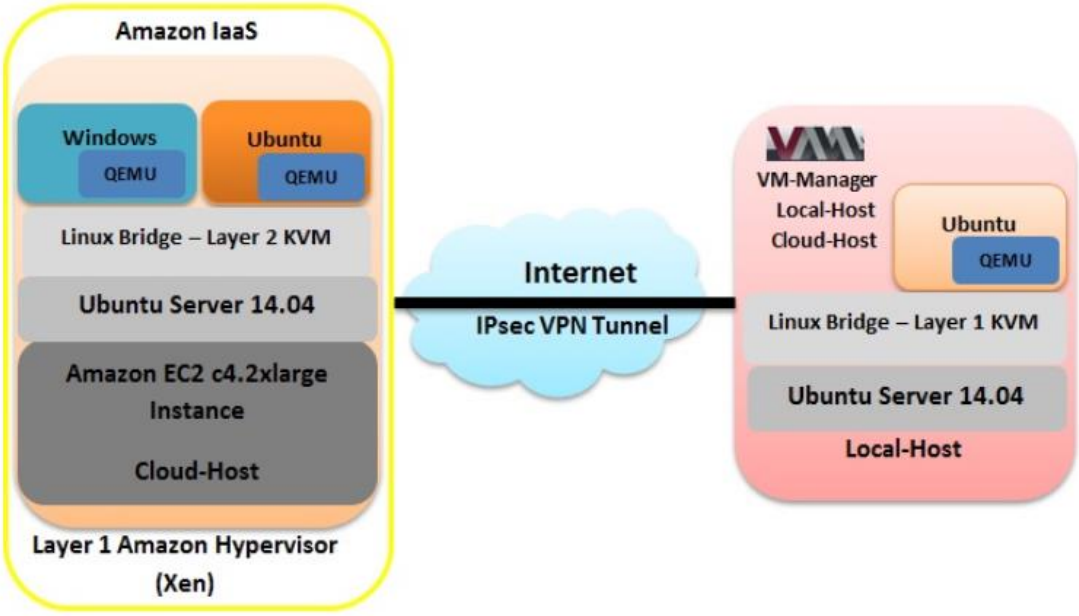


Figure 8 – LivCloud Architecture (I. E. A. Mansour et al., 2017)

Figure 8 represents the LivCloud architecture on Amazon EC2 and is designed base on live cloud migration criteria published on a research called "Interoperability in the Heterogeneous Cloud Environment: A Survey of Recent User-centric Approaches" (I. Mansour et al., 2016) that can be categorized in performance, flexibility and security. For performance, there are 3 criteria:

1. Live Migration must be imperceptible to the VM and to the users
2. Predicting the required resources to decide to proceed or not with the migration
3. Monitoring the resource utilization to avoid overutilization and to predict possible failures and problems with the migration

For flexibility, there are two criteria:

1. Decoupling the migrated VM from underlying system by supporting wide range of hardware drivers
2. Supporting different OS on the migrated VM

With respect to security, there are two criteria identified:

1. Maintain data privacy during all the live migration using encryption
2. Impose authentication during migration

One of the requirements of LivCloud was the support of nested virtualization in order do decouple VMs from the cloud IaaS and connect hypervisors on the IaaS in order to facilitate live migration back and forth. The paper explores in more detail about the migration and how to configure VPN specifications and virtual manager connections, explaining the idea and how this migration method works.

2.3.5 Spotify Migration – From On-Premise to one of the largest companies in the Cloud

Spotify, as many may know, is a Swedish music streaming company that was founded in 2006. In 2016, Spotify had over 1200 services, 20,000 daily job executions and hundreds of terabytes of data written daily(*Spotify Case Study*, n.d.). Over time, the maintenance and scalability of services became complicated, which led to the idea and possibility of migrating to the Cloud. For this migration to run smoothly, a detailed and tactical plan was developed to facilitate the work of developers and not affect their daily users.

Before any work to migrate components, Spotify worked using a prototype so they could build some guiding plans, solve some blockers and have a minimal working use-case to show to the rest of the teams (Google Cloud Tech, 2018). This stage was followed by Preparation, Migration Work and Maintenance Work.

2.3.5.1 Preparation Phase

This stage started the real migration plan for Spotify and was very important because in this forum the migration team was created. Made up of elements from Spotify and from Google, this team carried out a set of planning and preparation tasks so that the work of the product teams would be as fast and simple as possible.

There were five tasks that were the main focus of this team in this phase(Google Cloud Tech, 2018):

1. Build a visualisation of the migration state – the team developed a green/red circle migration visualisation team to understand the state of the components. That visualisation looked like a set of red (Spotify datacentre) and green (Google Cloud) bubbles, with each bubble representing a system, and the size of the bubble representing the number of machines involved. A visualisation example can be seen in the Figure 9 below.
2. Build a standardized migration sprint program – develop a general plan for the migration

3. Create teaching material for common migration cases – Teaching Material for specific and common cases like Java Programs, Apache Hadoop, Databases, between others (Google Cloud Tech, 2018).
4. Create teaching material for cloud – general material about how the cloud works, how to maintain the components deployed in the cloud and much more information.
5. Test all the work – tested all the material and tooling to verify that everything worked as expected

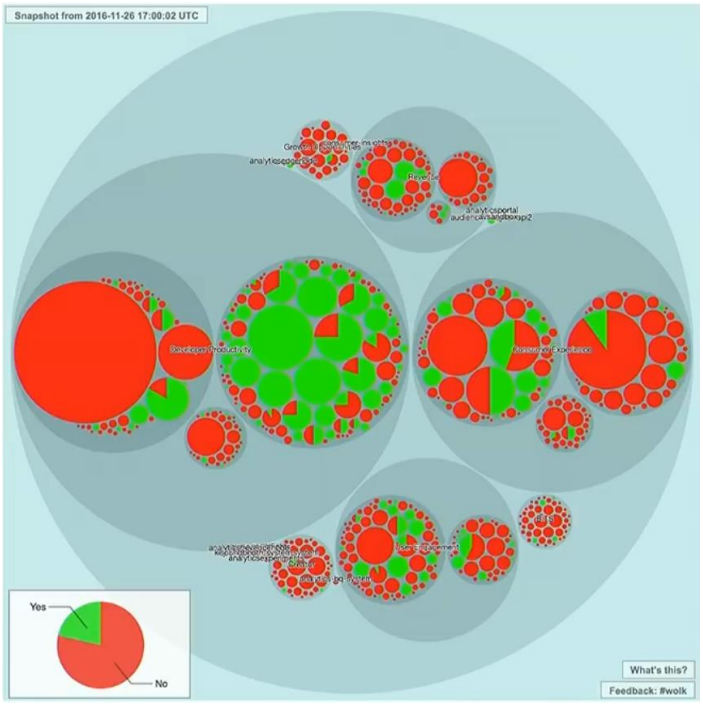


Figure 9 – Spotify Migration Visualisation (Google Cloud Tech, 2018)

Before starting the migration work, Spotify applied a technique that they called “Focus and Intake” (Google Cloud Tech, 2018). Focus was a phase when they would negotiate the sprint time in the product roadmap and it was called Focus because it had the objective of having all the team focused during one or two weeks of the spring just migrating components.

The Intake stage was mainly to understand how many components each team maintained, which components they would want to migrate and assess size and scope for each one. In this phase, they would also assess the Operational Risk and order the components by their size so they could tackle the biggest first.

2.3.5.2 Migration Phase

As mentioned above, Spotify engineering teams were tasked with moving their services to the cloud in a two-week sprint, where they paused any product development (Carey, 2018). During this phase, the work was divided in three parts:

- Migration Work (50% of the time)
- Reliability Testing (30% of the time) – dealing with incidents and test operating services
- Education (20% of the time) – study topics like Networking, Google Products and Support, Cost, between other topics.

This structure was used with all teams that migrated their services.

After getting the service running in the cloud, the migration of users were made gradually, from 5% to 100%, starting in October 2017 and finishing in the end of 2018 (Google Cloud Tech, 2018).

In this stage, there were two approaches they called Forklifting Path and The Rewrite Path. Both of them, were solutions to migrate services without needing to stop everything or copying data using cross jobs between on-premise to cloud and vice-versa using the VPN and adding load on the network.

- Forklifting Path
    - As the name implies, this method is about a team lifting and shifting their systems to Google Cloud. It was not the ideal solution but it was used in teams with little time
- The Rewrite Path
    - This method was used when teams didn't feel comfortable just porting over their jobs using the forklifting path and sometimes services were not prepared for scalability in the Cloud.
    - “The biggest thing with rewriting was it required a much larger time investment from teams and as engineers what usually happened is as they started writing it they want to rearchitect it too.” (Google Cloud Tech, 2018)

2.3.5.3 Maintenance Work Phase

As the name implies, this phase consists of the maintenance and support of the components. Since Spotify is using Google Cloud, the infrastructure part is supported by Google and Spotify's internal teams only need to support the components and application code.

2.4 Study Conclusion

After this study and technological analysis of the problem and the existing hypotheses in the market, it is possible to identify some important aspects and some existing flaws:

- It is very important to take into account the service and deployment models. Both models differ a lot depending on the needs of the company (deployment model) and the specific needs of a service (service model). As identified in several pre-migration phases of the migration models analysed, the choice of service and deployment models is very important because it decides the entire migration process as it influences how the services will be migrated (service model) and how the entire structure will be assembled within the company (deployment model).
- Moving between clouds not only affects services but the entire enterprise. Consideration must be given to what is gained and what is lost with this change. As referred to in Conway's Law "The Structure of Software will mirror the structure of the Organization that build it". As mentioned by author Michael Kavis in his book "Architecting the Cloud: Design Decisions for Cloud Computing Service Models" (Michael Kavis, 2014), the choice of a service and deployment model affects the entire structure of the company from its systems to customer support, finance, legal, sales and even human resources. In models such as Spotify, it is possible to see clear changes in its service and internal structure after this migration, as it allowed them to focus more on their product and sell it more quickly to the whole world.
- It is important to analyse each component in order to understand its needs and if migration can be straightforward or if refactoring is necessary. The vast majority of existing on-premise cloud components were designed taking into account the infrastructure where they would be deployed and, therefore, migration to public cloud is often not straightforward and requires changes in services.
- Never skip the planning step. An evaluation of the problem is mandatory and it is recommended to follow the rule: “Why? Who? What? Where? When? How?".
- We should never set expectations too high but realistically.
- Before a migration, everyone must be well informed about the subject and about the process.
- The cost is not always lower when using the public cloud. The entire contract and application operation must be taken into account to predict the cost
- None of the methodologies analysed responded to all needs, but of all of them, the plan used by Spotify presented the most complete methodology.

The entire analysis emphasized the need for a more complex methodology that addresses all the points mentioned above. A methodology will be developed based on the concepts referred to in this conclusion of the study, using ideas from the main market solutions analysed.

3 Value Analysis and Proposition

This section will focus on the value that will be proposed by the solution designed and developed in this dissertation. It will start by analysing the innovation process using the New Concept Development (NCD) Model, focusing more on the Fuzzy Front End and New Concept Development Stages. After this first analyse, a perceived value will be identified, followed by the definition of the value proposition.

In this chapter, some decision analysis methods such as AHP and QFD will also be used.

3.1 New Concept Development (NCD) Process Model

The process of developing a new product requires a systematic method to launch and commercialise it. The method that helps with this process is called New Concept Development Model.

New Concept Development process is described as a disciplined and defined set of tasks and steps that describe the normal means by which a company repetitively converts embryonic ideas into saleable products or services (Kenneth B. Kahn, 2005).

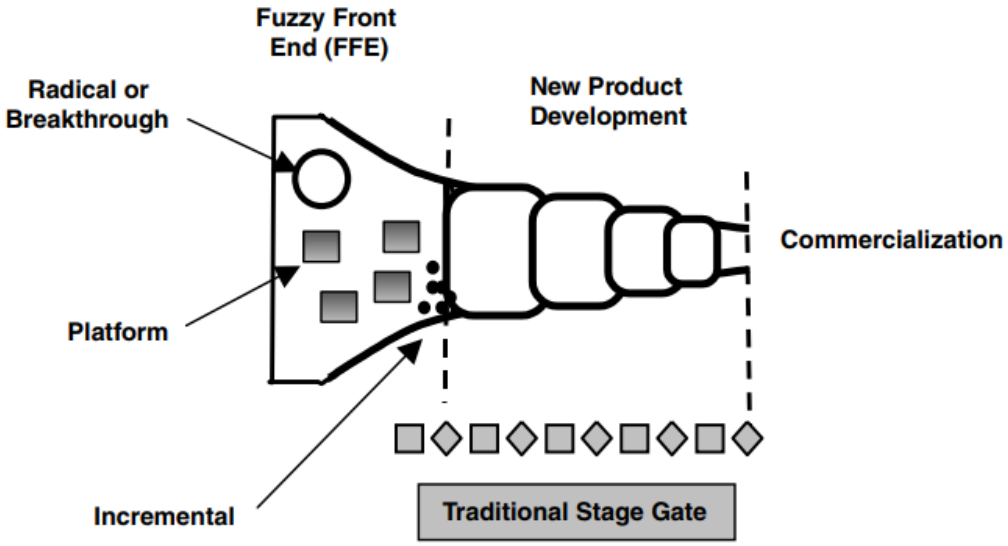


Figure 10 - Innovation Process Model

The innovation process may be divided into three different areas: the **Fuzzy Front End** (FFE), the **New Product Development** (NPD), and **Commercialization** (or Fuzzy Back End as described in some papers, that include Product Design and Testing) as shown in the image above(Kenneth B. Kahn, 2005). The main focus of the Fuzzy Front End phase is to identify the market opportunities and analyse different ideas and concepts before the real development and product work. After this stage, the New Product Development phase will focus on developing some of the filtered ideas from Fuzzy Front End until the product is delivered on the Commercialization Phase.

The NCD is not a linear process and can be described in three parts:

- the influencing factors
- the engine that drives the activities in the Fuzzy Front End
- five activity elements of the NCD

This can be represented by the model below:

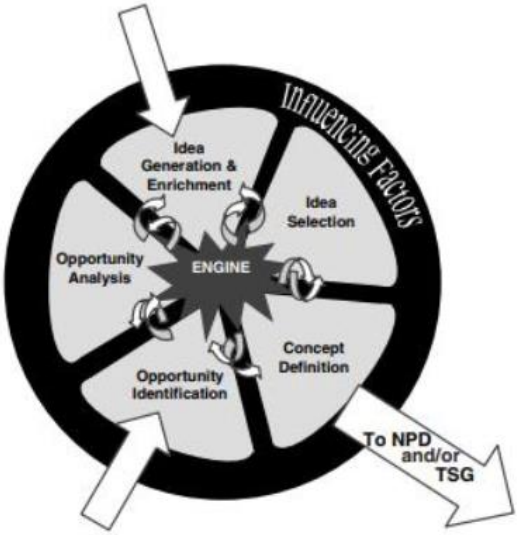


Figure 11 - NCD Model

This relationship model represents all the influencing factors that affect the entire innovation process and are driven by the engine represented in the middle of the model. This engine represents the factors that drive the five key elements that are controllable by the corporation, such as the leadership, culture, and business strategy.

Finally, the 5 highlighted elements are the main elements that are controllable by the organization: idea generation and enrichment, idea selection, opportunity analysis, opportunity identification and concept definition.

3.1.1 Opportunity Identification

In a constantly growing technological world, companies often have difficulty maintaining servers for their components and scaling them quickly to exceed customer needs. Over the years, more and more providers have appeared and grown in the public cloud provisioning market and providing services to support cloud development and maintenance.

Looking to the trend in the past 7 years, we can see the growing demand for public cloud services and research related to some of the largest providers in the world (AWS and Google Cloud Platforms):

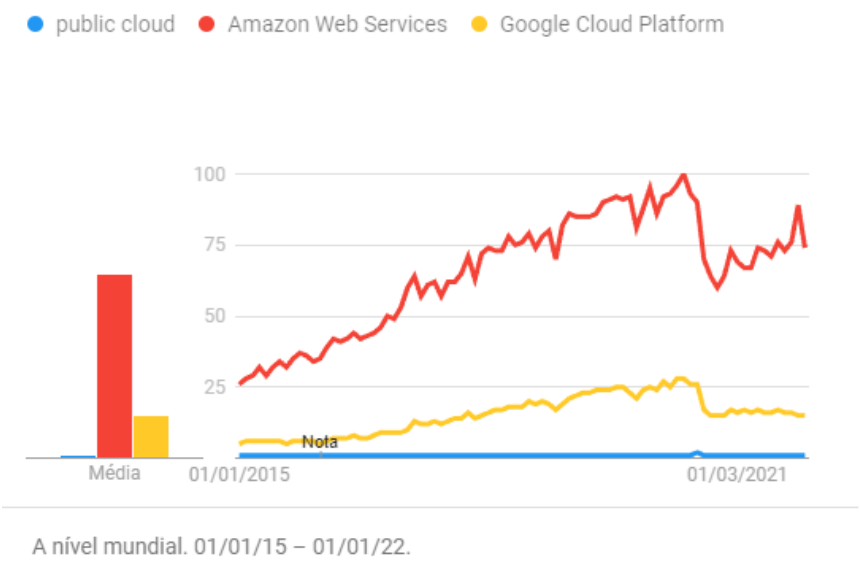


Figure 12 - Cloud Search Trend

By using Public Cloud to create a Full Public or Hybrid Cloud for a company, there would be some benefits:

- possibility to focus more on product development and implementation of new ideas
- have scalable and easily erasable test environments
- long-term cost effective and increased efficiency
- removal of incidents and concerns related to server or network problems (in the case of Full Public Cloud)
- Increased code quality with greater concern for scalability

3.1.2 Opportunity Analysis

The current stage of the NCD model attempts to examine the opportunity's potential and viability after introducing it. In most evaluations, market assessments, research, and statistical reports are utilized to better explain and contextualize the opportunity's worth.

Cloud Migration may be the solution and can give some benefits for a company, but it is necessary to do a further analyse and define the opportunity.

A survey performed by Synergy Research Group shows that the last quarter of 2021 had enterprise spending on the cloud infrastructure of around 45 million dollars, which represents 37% more than the same quarter of 2020. This growth was also accompanied by an increase in the attraction of large cloud service providers such as Amazon, Microsoft and Google, which increased their market share to 33%, 20% and 10% respectively (Synergy Research Group, 2021).

This shows the increase that is happening every year in the Cloud Infrastructure Services Market, regardless of the service model (IaaS, PaaS, SaaS) purchased by the companies.

Several large companies like Spotify made the change from on-premise servers to public cloud servers in the last 5 years, using different migration methods and, in the end, showed that the migration had more benefits than negative points. Migrating from their servers to the Google Cloud Servers, Spotify removed a lot of the operational complexity from their ecosystem, freeing up a lot of time for their team to focus on what mattered, that is, creating a product and satisfying the customer.

Despite this growth and these cases of large companies, many developers and companies (small, medium or large companies) show some fear when talking about this type of migration or change in their systems, due to fears related to security, data, lack of information, between others. For this reason, this type of document and clarification can be central and essential for some companies to change their system to something better for themselves and their customers.

3.2 Perceived Value

The link between perceived advantages and sacrifices is what is known as perceived value. The entire advantages must outweigh the sacrifices for a consumer to believe the solution is worthwhile.

To realize the value of the migration of systems from on-premise to cloud, it is necessary to identify the stakeholders and what they consider benefits and sacrifices:

- **Developers**:
    - **Benefits**
        - Developers can increase their productivity because they will focus more on their product instead of solving problems in the infrastructure or changing deployment configurations
        - More work about developing product and making the old systems more scalable, instead of infrastructure work
    - **Sacrifices**
        - Loss of control of the infrastructure and some parts of the system. Sometimes they can lose full control of a component if it is migrated to SaaS.
        - Learning curve sometimes large depending on existing systems and infrastructure
        - Migration work or adaptation of systems to the cloud can sometimes be time consuming

- **Company**
  - **Benefits**
    - The migration can be long-term cost effective because the company will not have to worry about servers and their maintenance or scalability
    - More Focused on the Product
    - More Scalable and reliable service, that will be useful in case of increased number of users
  - **Sacrifices**
    - Initial high cost of migration, adaptation of services and development of tools
    - Initial Planning and general change in the company due to this migration. The structure of the software should reflect the structure of the company, as previously mentioned.
- **Users**
  - **Benefits**
    - Better User experience is expected. Better infrastructure will reflect in a better service to the User and a more fluid experience when using the systems

3.3 Value Proposition

Before defining the value proposition its necessary to identify some topics like the product, gain creators, pain relievers, gains, pains, and customer jobs, thus developing a value proposition canvas.

The Value Proposition Canvas was originally developed by Dr Alexander Osterwalder as a framework for ensuring product and market harmony. It details the relationship between the two parts of Osterwalder's broader Business Model Canvas: Customer Segment and Value Proposition (Alexander Osterwalder & Yves Pigneur, 2010). The Value Proposition Canvas is formed by two building blocks called customer profile and value proposition. The first one is composed by three topics:

- Gains – benefits that customer expects and needs
- Pains – negative experiences, emotions and risks pointed by the customer
- Customer Jobs – functional, social and emotional tasks customers are trying to perform

Regarding the value proposition map, it is also composed by three topics:

- Gain Creators – description how the product or service create customer gains and how it will add value to the customer. Normally the gains creators respond to some of the gains expected by the user.
- Pain Relievers – description of exactly how the product or service alleviates customer pains.
- Products and Services –the products and services which create gain and relieve pain, and which create value for the customer

Having in mind the Value Proposition Canvas structure, it is possible to represent this thesis value proposition in the following Figure 13.

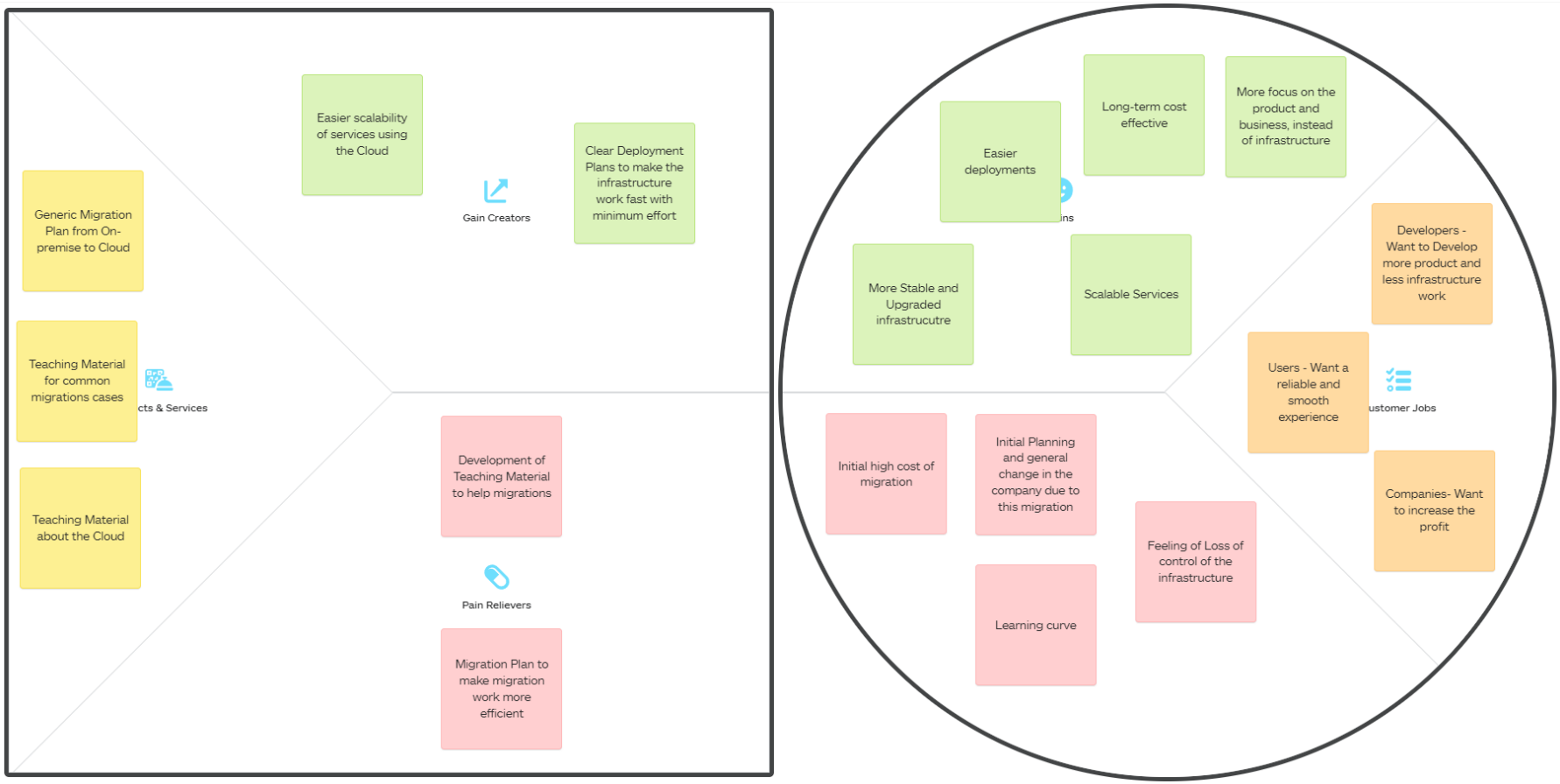


Figure 13 – Value Proposition Canvas (www.strategyzer.com)

This dissertation intends, as identified in Canvas, to present a solution to the problem of migrating services from on-premises to cloud, taking into account the pains and gains expected by the customer(s).

3.4 Quality Function Deployment (QFD)

Before applying this process to the problem under analysis, we must first understand what QFD is, what it is for and how it works.

Using the definition described by Quality-One International as a reference, "Quality Function Deployment (QFD) is a process and set of tools used to effectively define customer requirements and convert them into detailed engineering specifications and plans to produce the products that fulfil those requirements"(Quality-One International, 2017). QFD is a model developed in the 1960s in Japan and, since that, is being used to help putting customer needs first throughout the entire product development process. This way it guarantees that every technical need takes the consumer into consideration by consistently looping around to the Voice of the Customer, utilizing matrix diagrams such as the *House of Quality* to drive customer value into every stage.

To Implement QFD, a series of matrices are utilized in a 4-phase process to translate the Voice of the Customer to design requirements for each system, sub-system and component. The Four phases of QFD are (Quality-One International, 2017):

1. Product Definition - translating the customer wants and needs into product specifications. Also called The House of Quality because in this stage the House of Quality matrix is built (Quality-One International, 2017). Many organizations only get through this phase of a QFD process. Getting good data from the customer in this phase is critical to the success of the entire QFD process.
2. Product Development – in this phase, the critical parts and assemblies are identified and translated to critical or key part and assembly characteristics or specifications(Quality-One International, 2017).
3. Process Development - the manufacturing and assembly processes are designed based on product and component specifications
4. Process Quality Control – identification of critical part and process characteristics and development of appropriate process controls.

For this analyse, only the first Phase (Product Definition) will be used, taking into account the values obtained in the House of Quality matrix represented in Figure 14.

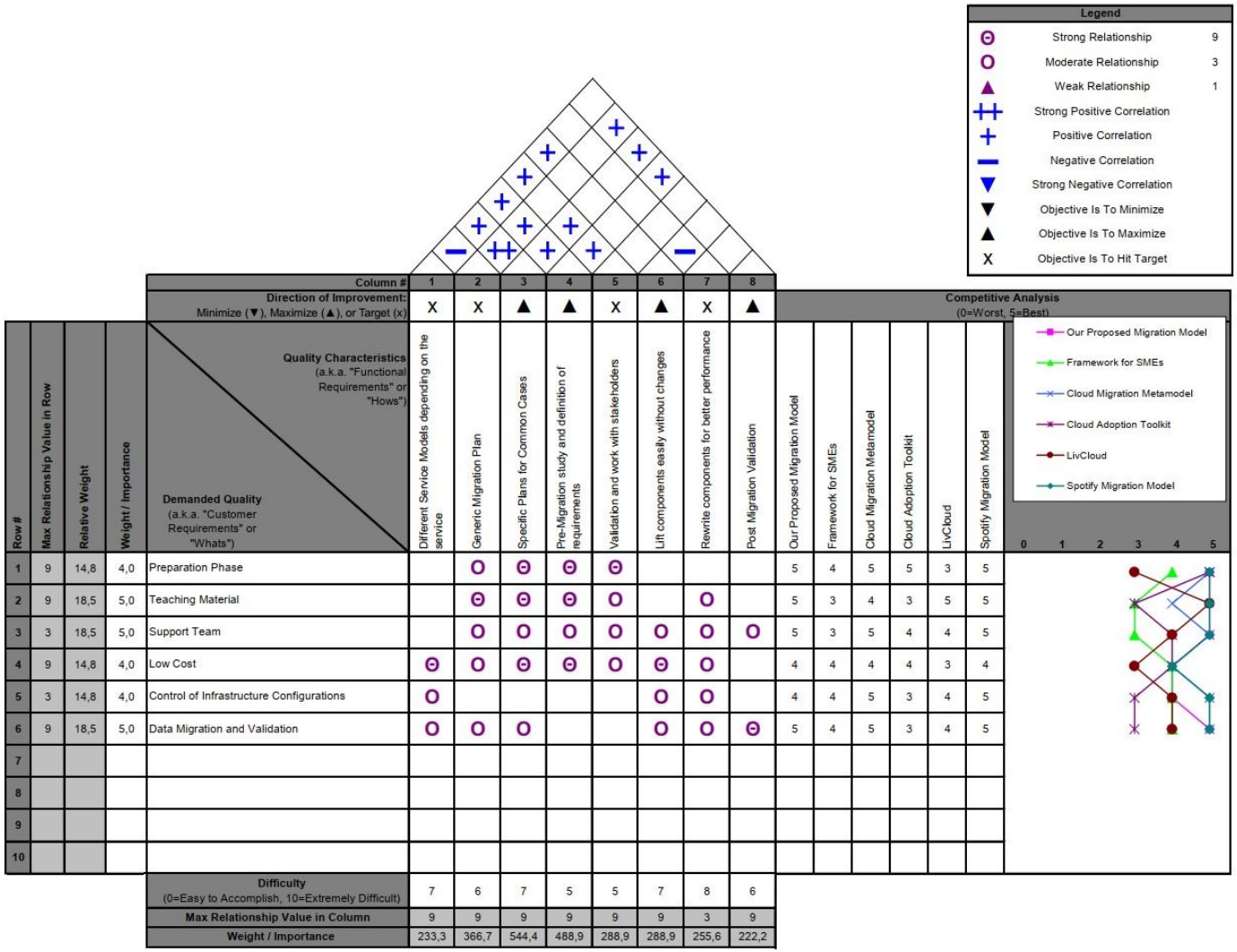


Figure 14 – House of Quality

3.5 Analytic Hierarchy Process (AHP)

Analytic Hierarchy Process is a system to solve complex decision-making problems, developed by Saaty in 1980 (Bouayad et al., 2018). AHP uses systematic evaluation criteria based on pairwise comparison and expert knowledge.

AHP aims to divide the problem in hierarchic decision levels, facilitating its comprehension, in three main operations:

- **hierarchy construction** - structure the decision hierarchy with the goal of the decision on the top, then the set of the criteria in the intermediate levels, and the set of Alternatives in the lowest level (Bouayad et al., 2018).
- **priority analysis** - construct a set of pairwise comparison matrices using the priorities defined by a fundamental scale
- **consistency verification** – check consistency and coherence of the judgement, using the Coherence Ratio (CR). CR is the division between Consistency Index (CI) and Random Index (RI).

3.5.1 First Step of AHP - Hierarchy Construction

As previously mentioned, the first step of the AHP method is to build the hierarchic decision three, with three levels representing the goal of the decision on top, then the criteria, and the alternatives. The hierarchy tree can be seen in Figure 15.

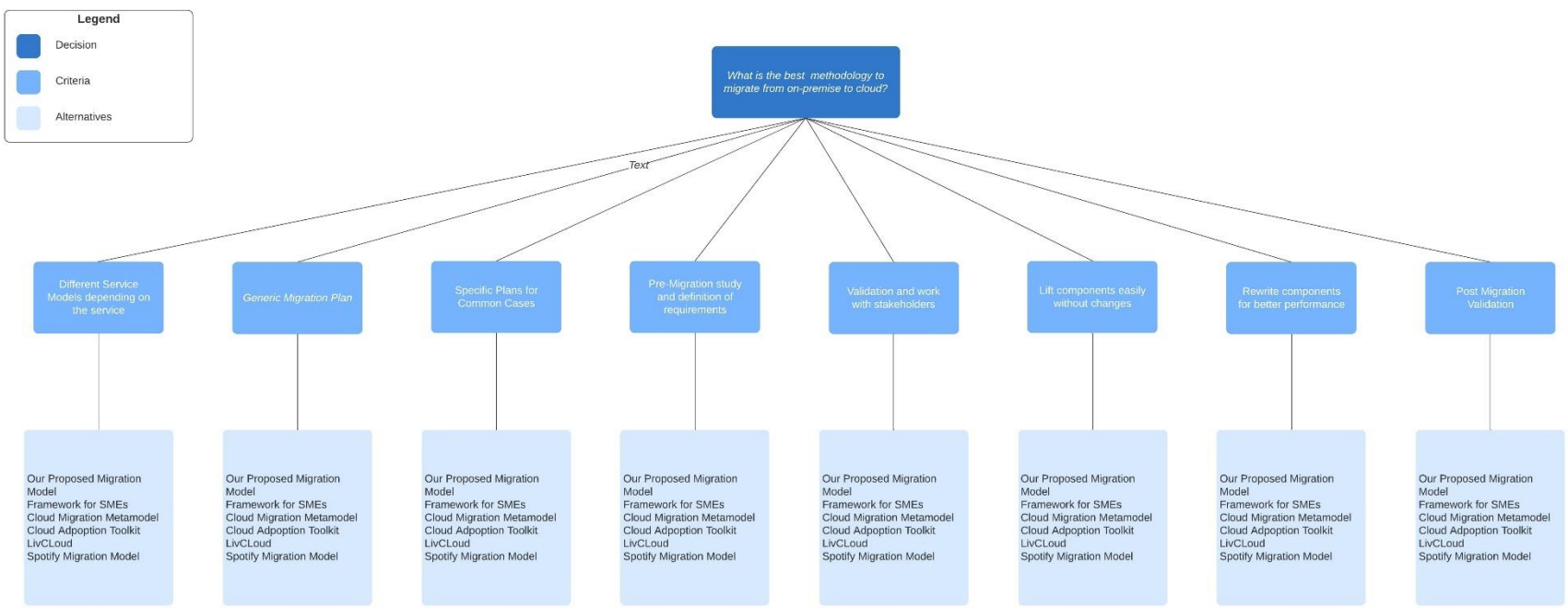


Figure 15 – AHP Hierarchy Construction

The problem and question that this proposition tries to answer is defined in the Level 1 of the Hierarchy, followed by the criteria in Level 2. For the criteria, there are 8 criteria:

- Different Service Models depending on the service
- Generic Migration Plan
- Specific Plans for Common Cases
- Pre-Migration study and definition of requirements
- Validation and work with stakeholders
- Lift components easily without changes
- Rewrite components for better performance
- Post Migration Validation

The level 3 represents the different alternatives available:

- The Proposed Migration Model
- Framework for SMEs
- Cloud Migration Metamodel
- Cloud Adoption Toolkit
- LivCloud
- Spotify Migration Model

3.5.2 Second Step – Priority Analysis

In the second step, a priority between the elements of each hierarchic level will be established, using a comparison matrix that can be represented on Table 4.

Importance Level	Definition	Interpretation
1	Equal Importance	Both activities contribute equally to the objective
3	Weak Importance	The experience and the judgment favor an activity over the other
5	Essential or Strong Importance	The experience and the judgment strongly favor and activity over the other
7	Demonstrated or Very Strong Importance	One activity is more favored than the other
9	Absolute Importance	The evidence favors one activity over ther other, with the highest degree of certainty
2,4,6,8	Intermediate Values	Intermediate values represent a compromise between two definitions

Table 4 – Fundamental Scale

The defined criteria pairwise comparison can be seen in Table 5.

	DiffServModels	GenerPlan	SpecPlan	PreMigSt	ValidStake	LiftEasy	RewriteCp	PostMigVal
DiffServModels	1,00	0,14	0,20	0,14	0,20	0,33	0,33	0,20
GenerPlan	7,00	1,00	5,00	3,00	3,00	3,00	5,00	3,00
SpecPlan	5,00	0,20	1,00	0,33	0,33	3,00	3,00	0,33
PreMigSt	7,00	0,33	3,00	1,00	3,00	3,00	3,00	3,00
ValidStake	5,00	0,33	3,00	0,33	1,00	3,00	3,00	0,33
LiftEasy	3,00	0,33	0,33	0,33	0,33	1,00	3,00	0,33
RewriteCp	3,00	0,20	0,33	0,33	0,33	0,33	1,00	0,33
PostMigVal	5,00	0,33	3,00	0,33	3,00	3,00	3,00	1,00
Sum	36,00	2,88	15,87	5,81	11,20	16,67	21,33	8,53

Table 5 – Criteria Pairwise Comparison

After obtaining the criteria comparison table, the next step is to obtain relative priority of each criteria, which is done by normalizing the comparison matrix and then obtaining the priority vector, by calculating the average of the values in each line of the normalized matrix.

	DiffServModels	GenerPlan	SpecPlan	PreMigSt	ValidStake	LiftEasy	RewriteCp	PostMigVal	Weights
DiffServModels	0,03	0,05	0,01	0,02	0,02	0,02	0,02	0,02	0,023945
GenerPlan	0,19	0,35	0,32	0,52	0,27	0,18	0,23	0,35	0,30093
SpecPlan	0,14	0,07	0,06	0,06	0,03	0,18	0,14	0,04	0,089785
PreMigS	0,19	0,12	0,19	0,17	0,27	0,18	0,14	0,35	0,201449
ValidStake	0,14	0,12	0,19	0,06	0,09	0,18	0,14	0,04	0,118776
LiftEasy	0,08	0,12	0,02	0,06	0,03	0,06	0,14	0,04	0,068383
RewriteCp	0,08	0,07	0,02	0,06	0,03	0,02	0,05	0,04	0,045869
PostMigVal	0,14	0,12	0,19	0,06	0,27	0,18	0,14	0,12	0,150863
Sum	1	1	1	1	1	1	1	1	

Table 6 – Normalized Matrix and Priority Vector

After obtaining the priority vector, it is necessary to carry out the consistency test for the criteria under analysis, following the steps:

1. **Calculate the Weighted Sum Value** – Multiple the Pairwise Comparison Matrix by the priority vector obtained above.

	Weighted Sum Value
DiffServModels	0,21
GenerPlan	2,77
SpecPlan	0,77
PreMigS	1,89
ValidStake	1,07
LiftEasy	0,57
RewriteCp	0,39
PostMigVal	1,41

Table 7 – Weighted Sum Value

2. **Calculate the intermediate vector** – the above weighted sum values are divided by the respective priority vector value:

	Weighted Sum Value	Priority Vector	Weighted Sum Value/Priority Vector
DiffServModels	0,21	0,023945	8,590
GenerPlan	2,77	0,30093	9,189
SpecPlan	0,77	0,089785	8,570
PreMigS	1,89	0,201449	9,384
ValidStake	1,07	0,118776	8,995
LiftEasy	0,57	0,068383	8,264
RewriteCp	0,39	0,045869	8,451
PostMigVal	1,41	0,150863	9,323

Table 8 – Intermediate Vector

3. **Calculate Self Value** - Adding all the values obtained in the previous step and dividing by the number of criteria:

$$\frac{8{,}590 + 9{,}189 + 8{,}570 + 9{,}384 + 8{,}995 + 8{,}264 + 8{,}451 + 9{,}323}{8} = 8{,}845673468$$

4. **Calculate Consistency Index**

$$CI = \frac{\lambda_{max} - n}{n - 1} = \frac{8{,}845673468 - 8}{8 - 1} = 0{,}120810495$$

5. **Calculate Consistency Ratio** – Consistency Ratio is obtained through the following formula, which divides the value of the consistency index by a value called random index, which is a tabulated value that differs depending on the number of criteria

$$CR = \frac{CI}{RI}$$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

Table 9 – Random Index Values for AHP

Using the value assigned in the table above for 8 criteria:

$$CR = \frac{0{,}120810495}{1{,}41} = 0{,}085681202$$

Since the calculated value is less than 0.1, it ensures the consistency of the criteria comparison matrix.

After creating a consistent criteria comparison matrix, the same process must be done on a comparison table for each criteria, considering the selected alternatives. Since this analysis generated several tables, the full AHP analysis can be seen in Appendix B – AHP Analysis.

After applying the same process for each criteria, an alternative composite priority table will be generated by multiplying the priority vector of each criteria's comparison matrix with the criteria's relative priority obtained in the previous criteria comparison matrix. This operation is represented in Table 10.

	DiffServModels	GenerPlan	SpecPlan	PreMigSt	ValidStake	LiftEasy	RewriteCp	PostMigVal
PropMig	0,3668	0,3615	0,3458	0,2479	0,3387	0,2571	0,3333	0,3069
SMEFram	0,1606	0,2075	0,1759	0,1994	0,0936	0,1963	0,0968	0,2025
Cloud Model	0,1039	0,0961	0,1294	0,1287	0,0652	0,1292	0,1753	0,0674
CloudToolkit	0,0502	0,0562	0,0532	0,0664	0,1863	0,0531	0,0528	0,0674
LivCloud	0,0286	0,0313	0,0383	0,0354	0,0283	0,0714	0,0355	0,0376
SpotifyMod	0,2900	0,2475	0,2574	0,3222	0,2879	0,2929	0,3062	0,3182

X

Priority Vector
0,023945
0,30093
0,089785
0,201449
0,118776
0,068383
0,045869
0,150863

=

0,318
0,182
0,104
0,075
0,036
0,286

Table 10 – Alternative Composite Priority Table

In this analysis, it was compared five existing frameworks with the expected tool that will be developed. Bearing in mind that the tool intends to be as complete as possible, it demonstrates a superior value compared to the others, but a superior value can be noticed in one of the existing frameworks (0,286 for Spotify Model Framework). The alternative with the highest relative priority represents the AHP's final result, in this case represented in Green in Table 10. The orange value represents the model that will be developed.

According to the results, the framework that is best suited to migrate services from on-premise to cloud is the Spotify Framework and also with a good result the framework for SMEs. The framework that will be designed and developed by this document, will try to take the best points from these two existing frameworks, reaching an optimal solution.

4 Analysis and Design

Regarding the design of the solution, as this dissertation is focused on research, planning and development of a methodology/framework for migrating components between private cloud and public cloud, an analysis will be carried out at the level of how a migration will be represented and possible steps of the methodology that will be designed.

4.1 Analysis

Before defining a design for the methodology, it is necessary to carry out an analysis of the use cases and the needs of each actor in the process. Figure 16 represents the use cases for three main actors of this process: the Engineer, the Architect and the Stakeholder. The Engineer will be responsible of developing and maintaining the service, the Architect will define the correct service and deployment model accordingly to the requirements from the stakeholders and according to the specifications defined by the engineers. The methodology should answer all the necessities of each actor.

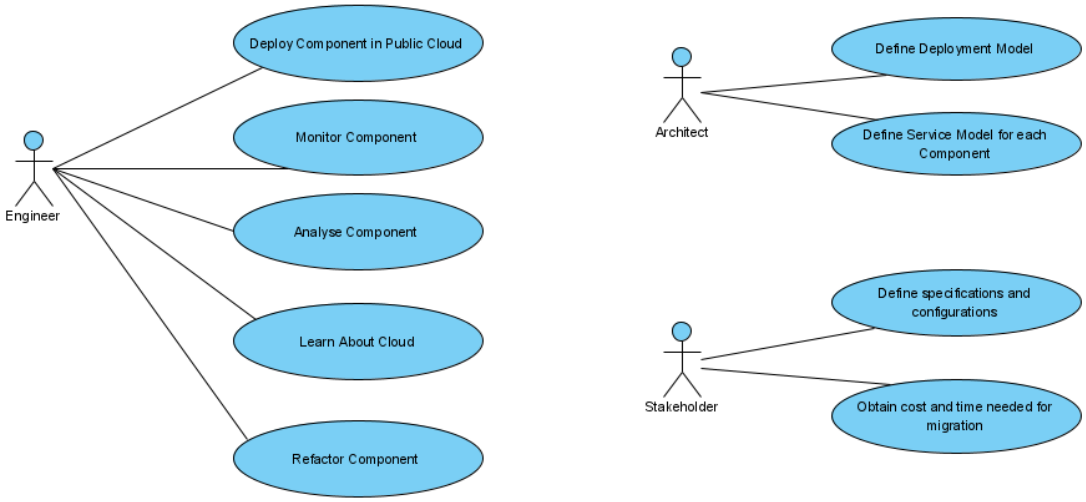


Figure 16 – Use Case Diagram

For the engineer, it is important that the methodology facilitates the deployment of components and allows him not only to learn more about the cloud itself, but also to analyse the component after migration so that there is certainty of the best possible performance. In the case of architects, their role is of great importance and will affect the entire development and migration process, so the methodology must help them choose the most accurate and reasoned choice of a deployment and service model for each component. As the company's representation role, stakeholders play a key role in the future smooth running of the entire company, defining and analysing acceptable specifications and costs for the migration and continuity of public cloud services.

Now that there is an idea of the cases that are needed for the process, it is possible to start defining the process that is the objective and idea of this project, Figure 17 represents a component that is in an on-premise cloud and that is intended to be migrated to a public cloud. The component migration and adaptation process correspond to the methodology to be designed and developed in this thesis.

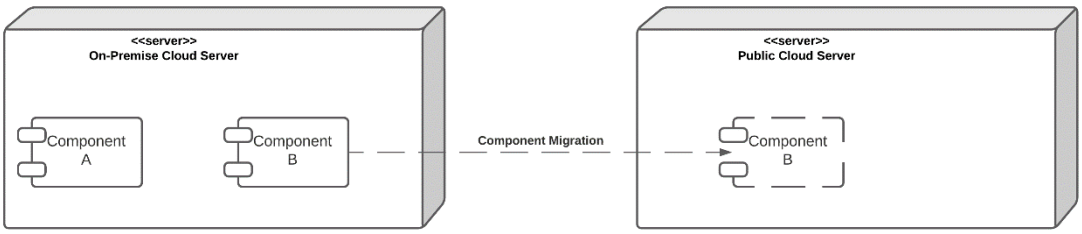


Figure 17 – Deployment Diagram

This migration process is not just about the deployment process on a new server but also includes a planning, testing and maintenance process.

In the Figure 18, a work flow idealized for the plan that will be developed is represented, involving the process from development to deployment, not leaving out the entire testing and maintenance process of the components. After the changes are committed, a package is generated that will have to go through several unit tests until it is ready for deployment. When the package is validated, it will be deployed in a development environment in the specified cloud (public or private), where it will undergo several integration and unit tests, some manual and others automatic.

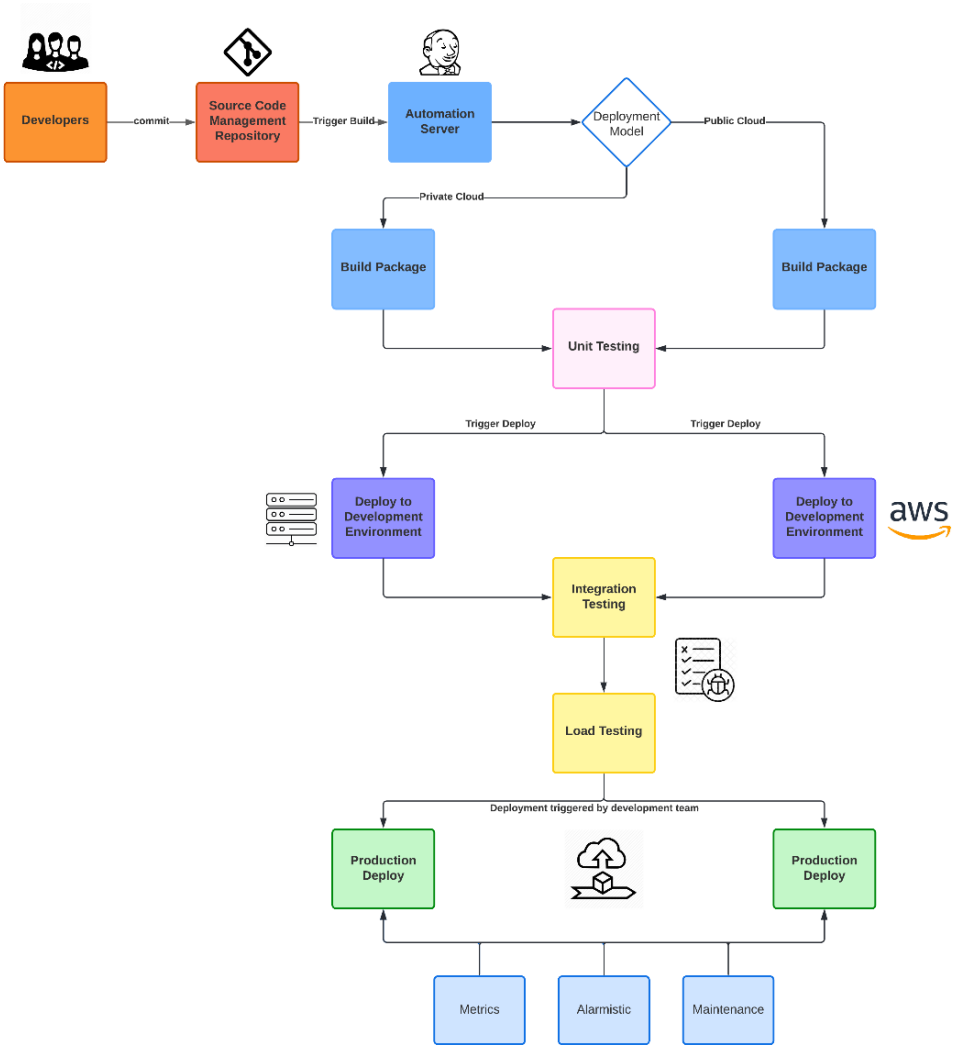


Figure 18 – Work Flow

4.2 Design

The main objective of this thesis is the development of a migration plan that will facilitate future migrations from on-premise cloud services to public cloud. The model idealized in Figure 19 is based on ideas from the various models analysed and evaluated in the Value Analysis, and is composed of three phases. The first stage is called Preparation Phase and will include the entire process of study, design, investigation and definition of models to be used. The second and most important phase is called Migration Phase and includes the entire flow from contacting stakeholders and defining specifications, to deploying and testing components in the new cloud. Last and not the Least, Maintenance Phase will be the final stage that includes all the continuous development and working in the migrated component, from monitoring and support to future system upgrades and improvements.

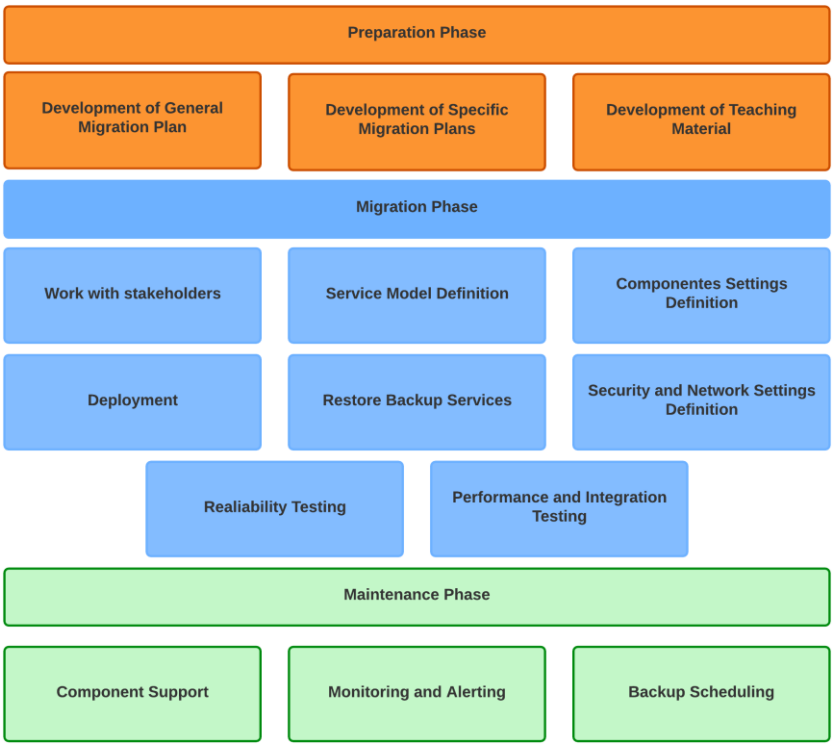


Figure 19 – Migration Plan Initial Design

This idea is still a draft of the General Migration Plan and will evolve during the development of the thesis. This one will be the basis for the development of more specific models depending on the technology or type of component in question, following stakeholder requirements.

There are some steps that can be moved like the “Work with Stakeholders” can be a preparation step instead of a migration step or even be an integrated phase in the two parts of the process.

4.3 Cloud Deployment Types Analysis

Before developing the migration methodology, it is also necessary to know more about Cloud and how it works. One of the most important points of this thesis is the deployment of a public cloud service and, therefore, it is an important point to analyse.

Being the project developed in a company with an active contract with a public cloud provider called Amazon AWS, this provider will be the same used in this project. For the implementation will be studies types of deployment of services in AWS.

Even before knowing the service to be worked on, it is essential to know the most used deployment methods in AWS being them: Amazon Elastic Beanstalk, Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS).

4.3.1 Amazon Elastic Beanstalk

Amazon Elastic Beanstalk is an easy-to-use service provided by Amazon to deploy and manage applications and services. This type of deployment stands out for the ease of integration with other AWS services such as databases and resources, the auto provisioning of the correct type of infrastructure and configurations taking the burden off the customer and the simple way to scale the application with requests distributed by an Elastic Load Balancer (ELB) (Morgan Perry, 2022).

For services where customers need more control over infrastructure and configurations, this type of deployment may not be the most appropriate choice. There are also some problems encountered in previous experiences with complex applications with different components like message queues, background jobs and a lot of microservices, which demonstrated that this type would not be suitable for that type of applications. Issues with deployments and automatic updates were also documented because large deployments can be very slow and difficult to troubleshoot in case of errors and Elastic Beanstalk keeps updating the underlying stack frequently and the client does not have any information about that until it breaks (Morgan Perry, 2022). This service also has the negative point of being a proprietary Amazon service, which makes future migrations to other providers more difficult.

Elastic Beanstalk makes easy for web applications to be quickly deployed and managed in AWS but the client needs to analyse well if this solution solves all the necessary requirements. The following figure shows a general use case for Elastic Beanstalk:

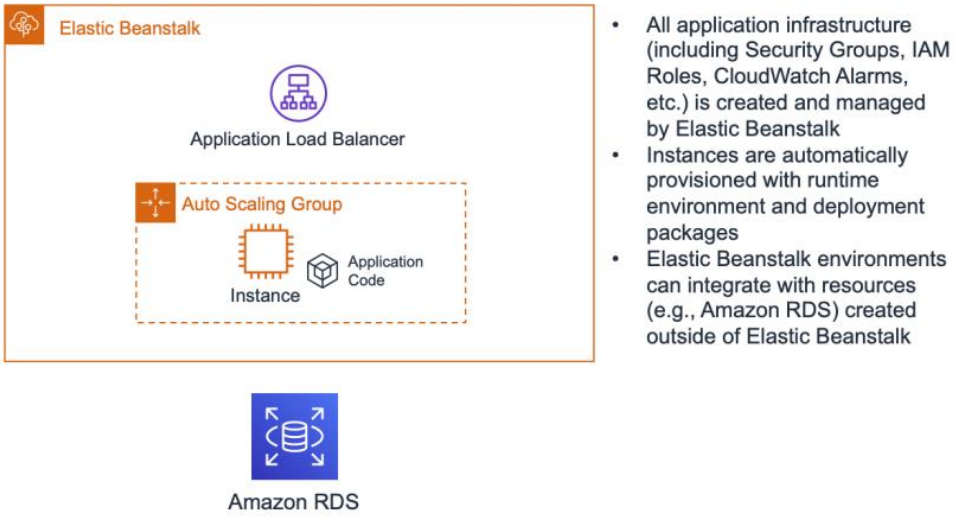


Figure 20 – Amazon Elastic Beanstalk (*Overview of Deployment Options on AWS - AWS Whitepaper*, n.d.)

Having in mind the advantages and limitations of Amazon Elastic Beanstalk, there are more solutions that can fit with client requirements, including Amazon Elastic Container and Amazon Elastic Kubernetes Services.

4.3.2 Amazon Elastic Container (ECS)

With the growing use of containers in the market, more deployment models for this type of service packaging have been developed, including Amazon Elastic Container (ECS). ECS is a fully managed container management service from AWS that supports Docker containers and allows users to easily run applications on a managed ECS cluster (Morgan Perry, 2022). The cluster can be formed by EC2 nodes or through serverless solution Fargate:

- Amazon Elastic Compute Cloud (EC2) is a service provided by Amazon that makes possible to run applications on the cloud, providing scalable computing capacity in the Amazon Web Services (AWS) Cloud. It eliminates the need to invest in hardware up front, so the client can develop and deploy applications faster. Users can use Amazon EC2 to launch as many or as few virtual servers as they need, configure security and networking, manage storage and scale up or down to handle changes in requirements or spikes in popularity (*What Is Amazon EC2?,* n.d.);
- AWS Fargate is a technology that can used with Amazon ECS to run containers without having to manage servers or clusters of Amazon EC2 instances. When using EC2 instances, client is responsible for the EC2 instances and needs to handle scaling, monitoring, patching, and security, in addition to the actual application concerns. Fargate looks a bit like a PaaS in that the user no longer needs to worry about the infrastructure work (*AWS Fargate*, n.d.).

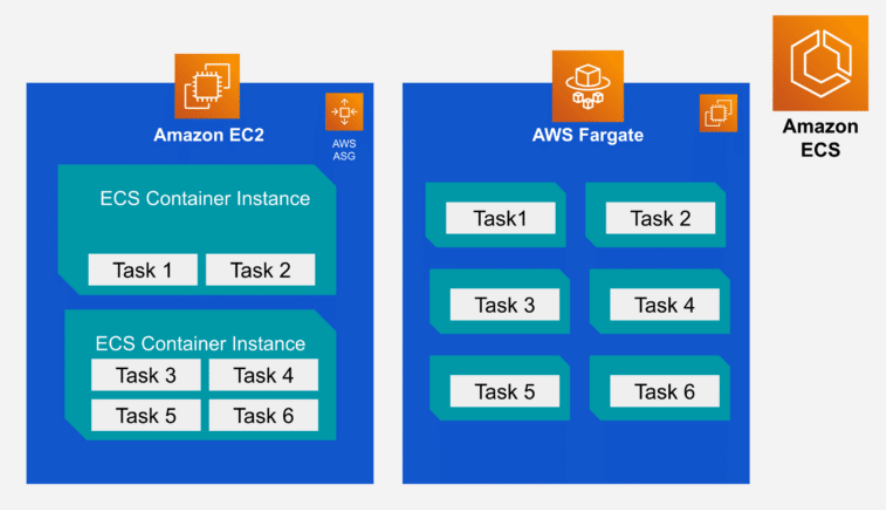


Figure 21 – Difference Between Amazon EC2 and Amazon Fargate (*AWS Fargate*, n.d.)

Figure 21 represents two ECS clusters with EC2 and Fargate instances and shows the differences described before.

ECS provide many advantages to the user by simplifying all the deployment process, from the infrastructure provisioning to the scalability and service management. When using ECS, all the infrastructure is provided and, if the user opts for using Fargate instances, all the management will be done by AWS. Scalability is also an important point in ECS that uses an elastic load balancer to distribute the work between instances and can incorporate auto-scaling groups as well. Being an AWS service, the integration with other AWS services like Pipelines and Databases are also possible and simple to implement.

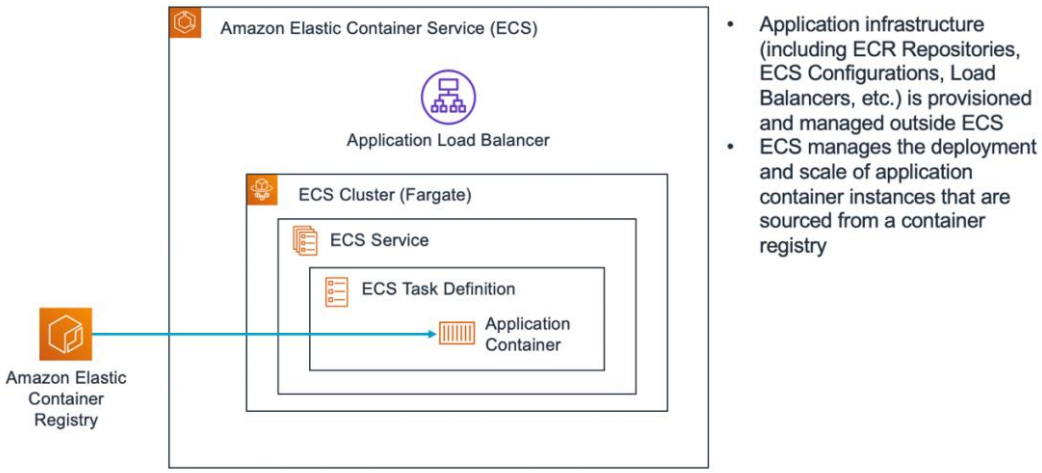


Figure 22 – ECS Cluster (*Overview of Deployment Options on AWS - AWS Whitepaper*, n.d.)

There also some limitations to point out like the no built-in load balancing (it uses AWS provided ELB), the existence of some difficulties to integrate with Elastic Container Registry (ECR) and to debug ECS logs using Cloud Watch for cases like a container going down or replaced. As referred before for Beanstalk, ECS is also not Cloud Agnostic and this can be important for applications that maybe moved to another cloud vendor in the future (Morgan Perry, 2022).

4.3.3 Amazon Elastic Kubernetes Service (EKS)

Amazon Elastic Kubernetes Service (EKS) is a service from AWS which helps running Kubernetes on the cloud without requiring the client to maintain their own Kubernetes Cluster or Control Plane. As with ECS, this service assumes the use of EC2 or Fargate instances.

Unlike the deployment services mentioned before, EKS is based on open-source Kubernetes and can therefore easily be migrated to another cloud provider in case of necessity. It also presents many advantages such as highly availability and scalability across multiple AWS availability zones and a built-in Load Balancing unlike ECS. Although the client can use AWS-provided Elastic Load Balancer, there is the option to run standard Kubernetes cluster load balancing or any Kubernetes supported ingress controller with the Amazon EKS cluster (Morgan Perry, 2022).

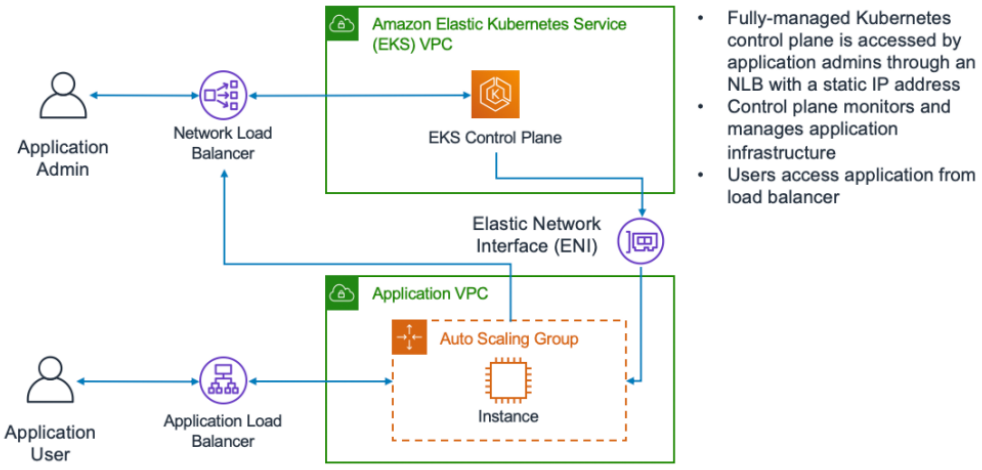


Figure 23 – EKS Cluster Representation (*Overview of Deployment Options on AWS - AWS Whitepaper*, n.d.)

Kubernetes is increasingly used in the market but it is still something new for many companies and therefore the learning curve for employees must be taken into account. Compared with other service deployment methods, EKS can be a little more expensive because there is an extra cost for every EKS cluster the user creates.

4.3.4 Best Deployment Type depending on the Service (Beanstalk vs ECS vs ECS)

Now that there is a clear idea of the most important types of Deployment in AWS, there is the necessity to understand when each solution should be used.

The best option is Elastic Beanstalk when most of the team members don't have the skills or knowledge to set an ECS or EKS cluster, if there is no strict need to manage the infrastructure and/or the team or the project comes from previous PaaS projects and want to do as few operations as possible.

For cases when the containerization is new to the team and the application isn't very complex and developed by a small team, Elastic Container Service (ECS) can be the ideal solution.

For more complex and large applications with a possible future implementation with multi-cloud or hybrid cloud, Elastic Kubernetes Service (EKS) can be the optimal solution especially if the team already support Kubernetes native applications.

	Amazon ECS	Amazon EKS
Open source	No - AWS proprietary	Yes - Kubernetes
Atomic Container Term	Task	Pod
Deployment Effort	Easy (AWS Dashboard)	Medium (AWS plus Kubernetes knowledge required)
Security (IAM)	Comes with the service	Requires addon software and additional configuration
Security (ENI support)	Yes - per task (single container)	Yes - per Pod (which can serve multiple containers)
Per VM Container Limit	Up to 120	Up to 750 Pods (which can host multiple containers)
System Service Cost	Used resources	Used resources plus ~$75 per cluster per month
Multi-cloud integration	No - AWS specific	Yes - Public and Private cloud integration.

Figure 24 – Amazon ECS vs EKS (Morgan Perry, 2022)

4.3.5 Analysis Conclusion

Before starting the implementation process, it is important to analyse the service and its requirements. The analysis was carried out taking into account the needs of the various stakeholders and the workflow.

After performing the problem analysis, an initial design of the migration model that is intended to be used was performed, followed by an analysis of the various types of deployment possible in the public cloud for a better understanding of the solution to be chosen in the next chapter. The choice of deployment model in AWS and the best solution will be chosen according to the analysis performed in this chapter.

5 Implementation

This section presents all the implementation and experimentation carried out taking into account the designed migration model and the cloud deployment types analysis made before.

Before starting the study and testing of the migration model, an introduction will be made about the framework that will be used to define the resources needed and an analysis of the service will be performed pre-migration.

The migration model will be tested using an existing service that is deployed in a private cloud and the current infrastructure and all its configurations will be documented further on.

5.1 Cloud Concepts

For this project, Amazon AWS will be the public cloud provider used to deploy all the necessary components. AWS was chosen by the company as a provider due to conditions such as price, scalability, the possibility of having different types of services deployed in various parts of the world and the existence of several ways to automate the deployment that are well documented.

Before starting with the migration and component study, there are some AWS concepts to be in mind such us AWS CDK that is the AWS framework used to define the application resources using different programming languages and the cloud deployment types analysed before in the *Cloud Deployment Types Analysis* chapter. These concepts will be further referred and used during the migration progress.

5.1.1 AWS CDK

AWS Cloud Development Kit (AWS CDK) is an open-source software development framework used to define cloud application resources with one of many programming languages supported such as *Typescript*, *Javascript*, *Python*, *Java* and *C#* (Amazon AWS, n.d.). CDK makes easier to create reliable infrastructure by creating secure defaults that can be extended and used by several applications.

For this case, it will be used Typescript because is the most documented and used language by AWS and the community.

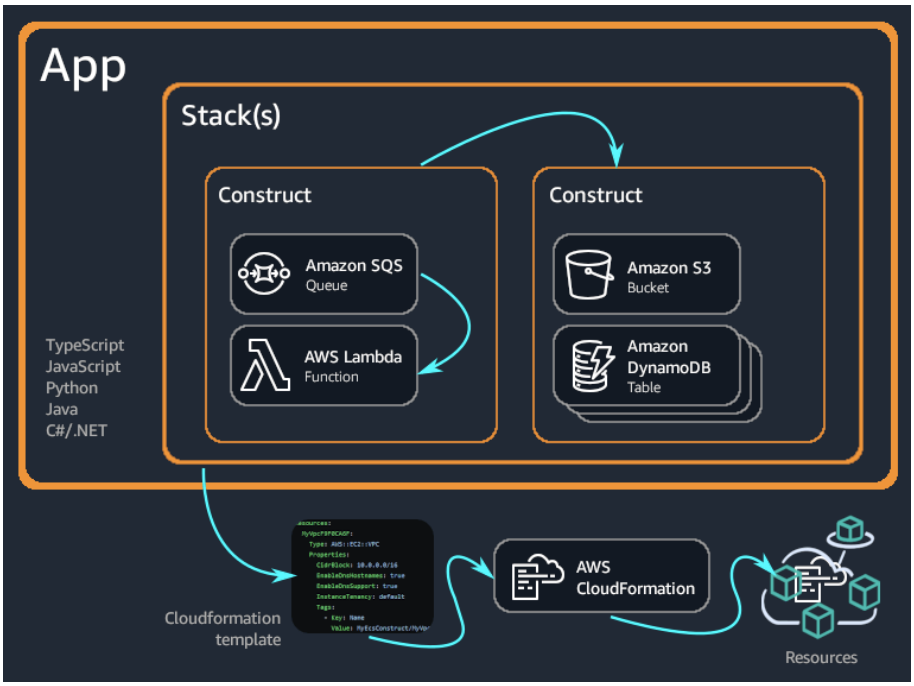


Figure 25 - AWS CDK and How it works (Amazon AWS, n.d.)

Every application will have a repository with its CDK code specifying all the resources needed that will be later compiled to a CloudFormation file that will be processed by AWS CloudFormation, generating/requesting all the necessary resources. A small CDK project with a few lines may generate more than 500 lines on the CloudFormation file template, making it easier to alter and avoid mistakes than manually constructing a CloudFormation template.

5.2 Initial Infrastructure and Deployment Process

Currently, the vast majority of company's services are in a private cloud infrastructure, which we call *i2*. This is maintained entirely by the company and is located in several datacenters in multiple locations. All the services are deployed in Virtual Machines and each one needs to have a package which is composed by several parts like the service itself, network configurations, number of VMs and VM configuration, security rules, between others.

As of today, all components deployed in the public cloud have been built from scratch and none have been migrated between clouds.

All the private cloud components are deployed using Pipelines and all the process is supported by company teams. On the other hand, public cloud components are configured using AWS CDK as the resource definition framework and Jenkins Jobs or AWS Code Pipelines are used to support deployment.

5.3 Migration Plan

The migration will be carried out according to the previously designed migration plan, with further development of each of the steps that complete it.

The Preparation Phase should be made by a specialized team with cloud knowledge and the main objective should be to develop a plan for the following phases, specifying steps to take to migrate each component from private to public cloud.

Figure 26 –Preparation Phase

In the current test, the company's DevOps team would be the right team to develop this phase, due to their knowledge of the internal infrastructure and public cloud, but for the development of this research, this work will be developed by the author of this dissertation.

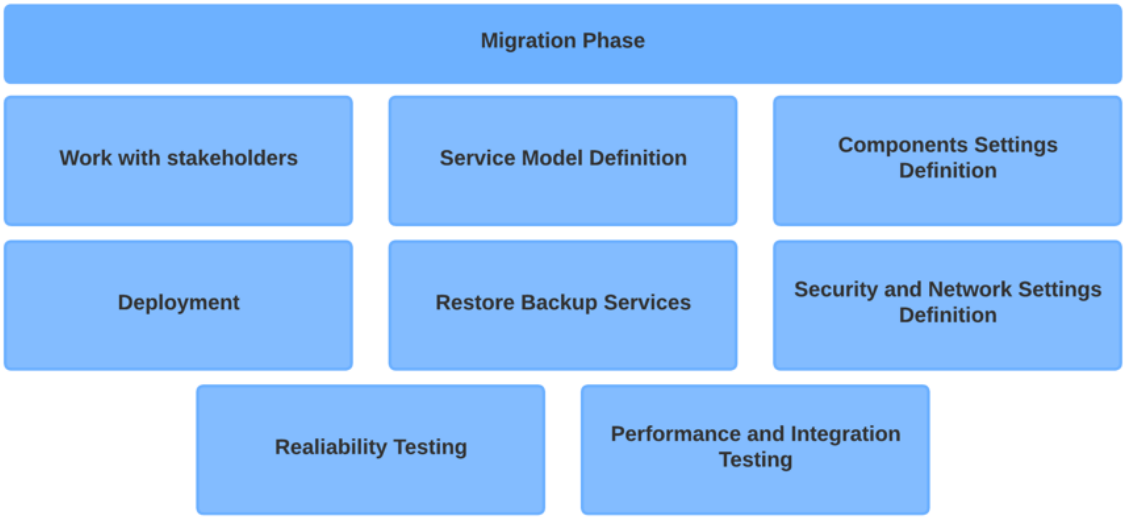


Figure 27 – Migration Phase

Having migration plans developed, the work will follow to the migration phase where the team responsible for supporting the migration will work closer with stakeholders to gather the most information possible about service and deployment configurations and all the security settings needed. This stage is also where decisions will be made such as what kind of deployment will be chosen and what are the best configurations for a better performance. The migration will conclude after running some tests to verify if the service is working properly and if the performance is as intended.

Figure 28 – Maintenance Phase

The phase after the migration will be the support phase, where there will be continuous evaluation and monitoring of the service, with support from the DevOps team in case of cloud issues.

5.4 Preparation Phase and Service Analysis

Now that there is a clearer idea of the migration process, it is necessary to know more about the Service to be migrated and analyse it so it's possible to select the best configurations and development type.

The service that will be used is a Spring Boot Application that will be called Spring Boot Service (SBS) for the purpose of this thesis. SBS is a *Kotlin* Application developed with *Spring Boot* and uses *Akka* Actors for concurrency data processing. This application is deployed in the company private infrastructure in development and production environments. This service has the objective of providing information about sports bets, giving the best possible bet opportunity for the client. For the service to work, there is a MySQL Database and an Apache Kafka cluster as requirements.

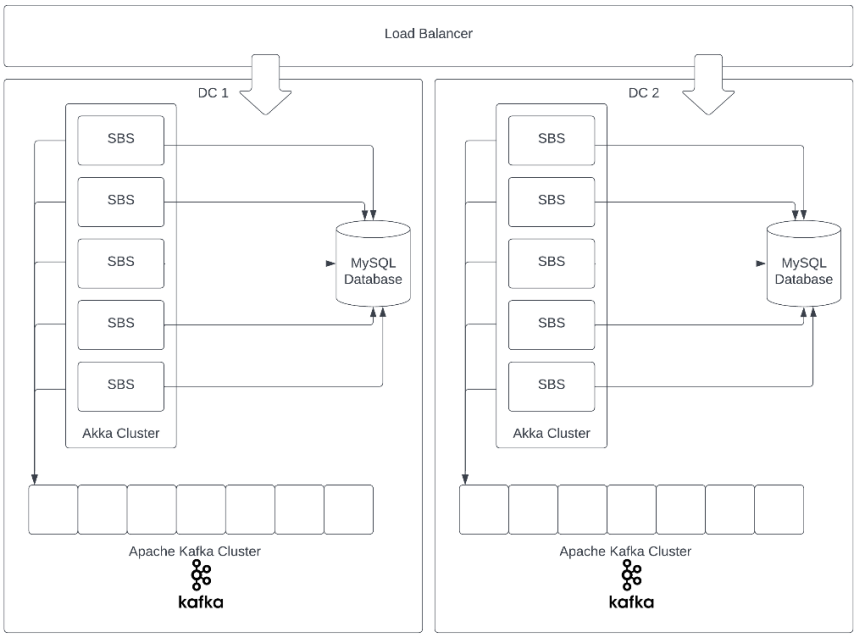


Figure 29 - Public Cloud Service Infrastructure

As shown in Figure 29, this component is deployed in two datacenters in the current private cloud and requires a MySQL database to read and write data and an Apache Kafka cluster to retrieve information from. This service validates and compares data from two different Apache Kafka topics, saving the information in the database and making it available through an HTTP endpoint.

5.4.1 Functional and Non-Functional Requirements

A very important point in the analysis of a component is to analyse its functional and non-functional requirements. As previously mentioned, this service is deployed in two datacenters and uses an active/active strategy for processing requests simultaneously by both datacenters. This requirement also includes a failover plan that is guaranteed by the use of a DNS Load Balancer that allows traffic to be shifted to only one of the datacenters in case of problems.

For Logging and Traceability, there are some application metrics exposed and distribution tracing is implemented. Metrics are collected using a custom collector that exposes them to a *Grafana* Metric Dashboard and all the Logs are also collected using a *Splunk* Collector which allows us to review logs during a contractually defined period.

Figure 30 – Grafana Labs Metrics Dashboard (*Grafana*, n.d.)

It is also important to mention that the location of the data should be taken into account, promoting the reading of data from the same datacenter for lower latencies and better service performance.

In the current infrastructure, the service is deployed in 3 different environments: a quality assurance environment, a performance testing environment and a production environment. The quality assurance environment will have lower system specifications and the performance environment will have specifications identical or equal to production for a near real environment performance test.

5.4.2 Points to consider before performing the migration

Before starting the migration there are some points to be in mind related with the infrastructure and the service itself:

- With this migration, there will be a change from Virtual Machine Deployment to Container Deployment and some services need to be adapted to do so. For this service, we will need to create a Docker Image before the deployment.
- Since this service uses a MySQL database, it is necessary to know if it is essential to copy the existing data to a new database in AWS or if the service can start with a clean database. Another point of view may be to keep the database in the Private Cloud in case of sensitive information which the company might not want to move to the Public Cloud.
- Caution should be taken with all aspects related to Security and Networking as communication ports between components will be required and there should be no connection from outside the company to this service
- Apache Kafka cluster will be needed and can be deployed with a SaaS option provided by Amazon called Amazon MSK (Amazon Managed Streaming for Apache Kafka). This option will remove all the infrastructure and maintenance work required to support a normal Kafka Cluster. The customer only needs to specify some settings and topics required and Amazon will be responsible for the rest of the work

5.5 Model Implementation

Now that the analysis is complete, the implementation process may begin by gathering input from stakeholders and developing infrastructure as code (*IaC*).

5.5.1 Stakeholders Meeting

These sessions are held with the goal of gathering information about the service and developing more detailed specifications to address some of the concerns raised above, such as MySQL database data and network and port configurations. All meetings must be documented, including a list of attendees, questions asked, and responses received.

The meeting record in Appendix C – First Stakeholders Meeting Record, answers the questions mentioned above in the points to consider. In this meeting, repositories and documentation regarding the service and how it is implemented and deployed were gathered and some doubts were discussed with the stakeholders that allowed establishing some actions such as the development of CDKs for the various components and the non-need for data migration from the database. With this information, it is possible to proceed with the development of the code that will create the services infrastructure and make the necessary changes to the *SBS*.

Later during the development of the infrastructure code, some issues arose regarding the service and the logging and metrics process. A new meeting documented in Appendix D – Second Stakeholders Meeting Record was held drawing the following conclusions: the initial deployment will use Cloudwatch to export the logs and after the service is running correctly, the development team will switch to using Grafana Cloud to configure custom metrics and log filters.

5.5.2 Infrastructure as Code (IaC)

As mentioned earlier, the project will use the AWS CDK as the framework for defining the required infrastructure, starting by developing CDKs for each of the core service dependencies.

Apache Kafka will need a CDK creating the Amazon MSK service, connected to the company virtual private cloud (VPC). The Amazon MSK solution will remove all the cluster maintenance work and the users will only have to worry about creating the necessary topics and connecting to get or produce data. We opted to use Amazon MSK as it is a provider offered solution that makes all the configuration and upgrades of an Apache Kafka cluster quite abstract.

CDK Code will have two main folders: *bin* and *lib*. In the bin folder, there is a main class that represents the App that will be created in AWS with the respective choice of deployment method that can be ephemeral deployment or a more complex option like *AWS CodePipeline*. Ephemeral is a simple deployment that will only create an environment for testing purpose that will be deleted in a short period of time. On the other hand, CodePipeline should be used to define stages and different environments from development to production. For a simpler approach, this migration will start by using only an ephemeral deployment, addressing the pipeline development approach at a later stage.

The structure of the files and the organization of the code will be done with special care because it is very important to have the same structure for all the repositories to make it easy to maintain these services in the future and to serve as an example for other similar services.

The first step when developing a CDK Project it's to run the "cdk init" command to generate the base structure for the project. It is necessary to specify the template to be used: *app* to create an empty CDK or *sample-app* to create a CDK with a base stack containing the deployment of an Amazon SQS queue and an Amazon SNS topic. The language is also important in this command. AWS CDK support multiple languages but, for this project, Typescript will be the language used, as it is the most widely used and supported language by the community and AWS.

The base project structure will essentially be as follows:

- A *bin* folder that contains the main class of the cdk project that will start all the stack creation
- A *lib* folder with all the specific environment configuration code and all the base stack configurations
- A *test* folder that will contain some tests for the stack creation
- A *package.json* file with all the necessary dependencies
- A *package-lock.json* will all the dependencies and versions locked for the project
- A *cdk.json* file that specifies what the *cdk* command will need to run (app) and what it is included or excluded like *readme* files, *node_modules*, between others.

Having all the base structure generated, it becomes easier to develop different projects changing only the stack generated by each one.

With all the structure defined, the code development can begin, initially creating the support services such as the Apache Kafka cluster and the MySql database.

5.5.2.1 Apache Kafka

The main class for the Apache Kafka infrastructure will only have the service name and the cdk App creation, calling the code that will create the test stack. In this class, it is possible to use code that allows the deployment of a test environment or a pipeline using *CodePipelines* that enables the creation of different pipelines for each environment. For this test, it will only be used a test environment and the *CodePipeline* alternative will be explored later in this document.

```
import 'source-map-support/register';
import * as cdk from 'aws-cdk-lib';
import {TestEnvDeploy} from '../lib/environments/test-env';

const tla = 'thesiskafka';
const app = new cdk.App();

new TestEnvDeploy(app, `ephemeral`);
```

Figure 31 - Main CDK class

The main class will call a *lib* class that will define the stack and the configurations needed. For this project the lib folder structure will be the following one:

- *environments* folder with one file for each environment
- *cdk-stack* file with the stack that will be called by each of the environments

To represent the test environment, the following class in Figure 32 will create the Apache Kafka Stack specifying some stack properties like service name, virtual private connection name, Apache Kafka version and configurations. In this step, it is also important to define security groups, specifying ingress and egress rules with the necessary ports. For this case, all the security rules will be opened to the company network IP range.

```
export class TestEnvDeploy extends Construct {

    constructor(app: cdk.App, env: string) {
        super(app, `${app}-${env}`);

        const securityGroupRulesDev = (sg: SecurityGroup): void => {
            // Ingress
            // VPC
            sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(9092)); // Kafka Brokers in plaintext
            sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(2181)); // Zookeeper default

            // Egress
            // VPC
            sg.addEgressRule(Platform.prefixList('ppb-network'), Port.tcp(9092)); // Kafka Brokers in plaintext
            sg.addEgressRule(Platform.prefixList('ppb-network'), Port.tcp(2181)); // Zookeeper default
          };

          new ThesisKafkaCdkStack(app, `thesisKafka-${env}-stack-quintap3`, {
            tla: 'thesisKafka',
            vpcName: 'dev',
            deploymentEnv: env,
            kafkaVersion: '2.7.0',
            numberOfBrokerNodes: 3,
            enhancedMonitoring: 'PER_TOPIC_PER_BROKER',
            clientBrokerEncryption: 'TLS_PLAINTEXT', // TLS, TLS_PLAINTEXT, or PLAINTEXT
            brokerProps: {
              instanceType: 'kafka.t3.small',
              brokerAzDistribution: 'DEFAULT',
              storageVolumeSize: 10,
              addSecurityGroupRules: securityGroupRulesDev,
            },
          });
    }
};
```

Figure 32 - Environment Infrastructure

The last and the most important step is the development of the CDK Stack, in this case, the Apache Kafka stack that will be implemented using AWS MSK. To create the Stack, a class is created that extends the class *cdk.Stack* provided by AWS in which you can configure and use custom properties (*StackProps*) as is the case in this implementation. To better configure the Apache Kafka cluster, an implementation of *StackProps* represented in Figure 33 was created with service specific settings and Apache Kafka specific configurations such as number of brokers, version and security rules.

```
interface ThesisStackProps extends cdk.StackProps {
  tla: string,
  vpcName: string,
  deploymentEnv: string,
  kafkaVersion: string,
  numberOfBrokerNodes: number,
  enhancedMonitoring: string,
  clientBrokerEncryption: string,
  brokerProps: KafkaBrokerProps
}

interface KafkaBrokerProps {
  instanceType: string,
  brokerAzDistribution: string,
  storageVolumeSize: number,
  addSecurityGroupRules: (sg: ec2.SecurityGroup) => void // This is a function signature. readonly
It modifies a security group.
}
```

Figure 33 - Apache Kafka Stack Properties

These props will be used to create an AWS MSK Cluster that will be represented as a *CfnCluster* object. *CfnCluster* will need some settings related to Apache Kafka and some of them can be passed by an external object called *BrokerNodeGroupInfoProperty*. The Figure 34 represents the creation of this same cluster and some of the configurations that were defined for this test.

```
export class ThesisKafkaCdkStack extends cdk.Stack {

  private readonly tla: string;
  private readonly vpc: ec2.IVpc;
  private readonly deploymentEnv: string;
  readonly cluster: CfnCluster;

  constructor(scope: Construct, id: string, props: ThesisStackProps) {
    super(scope, id, props);

    this.tla = props.tla;
    this.vpc = Platform.vpc(this, 'VPC', props.vpcName);
    this.deploymentEnv = props.deploymentEnv;

    const sg = this.getSecurityGroup();
    props.brokerProps.addSecurityGroupRules(sg);

    //2. Apache Kafka Cluster Deployment
    const brokerNodeGroupInfo: CfnCluster.BrokerNodeGroupInfoProperty = {
      brokerAzDistribution: 'DEFAULT',
      instanceType: 'kafka.t3.small',
      storageInfo: { ebsStorageInfo: { volumeSize: 30 } },
      clientSubnets: this.vpc.privateSubnets.map(s => s.subnetId),
      securityGroups: [sg.securityGroupId],
    };

    const kafkaCluster = new CfnCluster(this, 'Cluster', {
      clusterName: 'appThesisKafka',
      kafkaVersion: '2.7.0',
      numberOfBrokerNodes: 3,
      enhancedMonitoring: 'PER_TOPIC_PER_BROKER',
      brokerNodeGroupInfo,
      encryptionInfo: {
        encryptionInTransit: {
          clientBroker: 'PLAINTEXT',// TLS, TLS_PLAINTEXT, or PLAINTEXT
        },
      },
    });
  }

  private getSecurityGroup(): ec2.SecurityGroup {
    const description = `Security group for ${this.tla}-${this.deploymentEnv}`;
    return new ec2.SecurityGroup(this, `${this.tla}-SecurityGroup-${this.deploymentEnv}`, {
      vpc: this.vpc,
      description: `${description}`,
      allowAllOutbound: false,
    });
  };
}
```

Figure 34 - AWS MSK Cluster

After getting all the configurations, the project should be deployed using 'cdk deploy' command. The entire deployment process can be seen and analysed in the CloudFormation interface as represented in Figure 35.

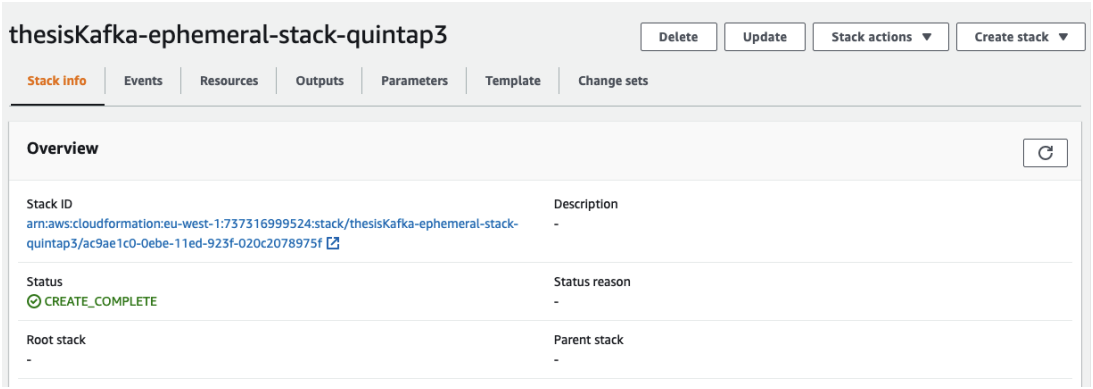


Figure 35 - CloudFormation MSK Cluster Information

With the cluster deployed, it is possible to access information about the cluster state and connections using the AWS MSK interface represented in the Figure 36.

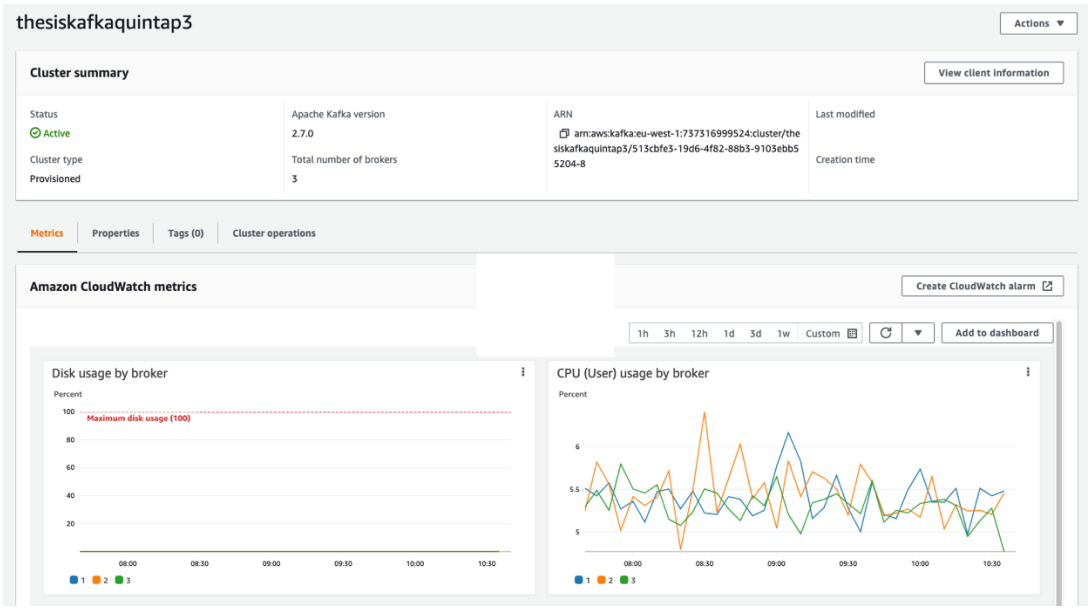


Figure 36 - MSK Cluster Configurations

Once the Apache Kafka cluster is active and healthy, the development of the remaining components can continue. The following component to be deployed is the Relational Database needed for the main service.

5.5.2.2 Relational Database

For the relational database, the CDK code structure is very similar to the structure used in the MSK Cluster CDK and will only vary in the stack creation. In the Figure 37 below, there is the code used to define the security groups and the stack creation. For Relational Databases, there is only the need to open ports for the database port (3306) and for ssh access (22).

```
export class TestEnvDeploy extends Construct {

    constructor(app: cdk.App, env: string) {
        super(app, `${app}-${env}`);

        const securityGroupRulesDev = (sg: SecurityGroup): void => {
            // Ingress
            // VPC
            sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(3306), 'ppb-network');
            sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(22), 'ppb-network'); //Allow SSH connections

            // Egress
            // VPC
            sg.addEgressRule(Platform.prefixList('ppb-network'), Port.tcp(3306)); //MySql Database
          };

          new ThesisDatabaseCdkStack(app, `sbs-${env}-stack`, {
            tla: 'sbs',
            vpcName: 'dev',
            deploymentEnv: env,
            securityGroupRules: securityGroupRulesDev,
          });
    }
}
```

Figure 37 - Relational Database Stack Creation

Following this environment file, there is a different stack from the MSK Cluster that will be defined using AWS RDS dependency. Figure 38 represents the stack creation with a generated

secret and the relational database instance definition. For the database a password is required for access and for this AWS Secret Manager was used to generate a password to access it. The settings used to create the instance are shown in the image and will be changed in the future depending on the environment and service needs.

```
export class ThesisDatabaseCdkStack extends Stack {

  private readonly tla: string;
  private readonly vpc: ec2.IVpc;
  private readonly deploymentEnv: string;

  constructor(scope: Construct, id: string, props: ThesisStackProps) {
    super(scope, id, props);

    this.tla = props.tla;
    this.vpc = Platform.vpc(this, 'VPC', props.vpcName);
    this.deploymentEnv = props.deploymentEnv;

    const sg = this.getSecurityGroup();
    props.securityGroupRules(sg);

    //4. MySQL Database - Generate Password and save it on AWS Secret Manager
    const mySQLPassword = new secretsmanager.Secret(this, 'DBSecret', {
      secretName: "SpringbootDB-DBPassword",
      generateSecretString: {
        ...
      }
    });

    const mySQLRDSInstance = new rds.DatabaseInstance(this, 'mysql-rds-instance', {
      engine: rds.DatabaseInstanceEngine.MYSQL,
      instanceType: ec2.InstanceType.of(ec2.InstanceClass.T2, ec2.InstanceSize.SMALL),
      vpc: this.vpc,
      storageEncrypted: true,
      multiAz: false,
      autoMinorVersionUpgrade: false,
      allocatedStorage: 10,
      storageType: rds.StorageType.GP2,
      backupRetention: cdk.Duration.days(3),
      deletionProtection: false,
      credentials: rds.Credentials.fromSecret(mySQLPassword),
      databaseName: 'thesisdb',
      securityGroups: [sg],
      port: 3306
    });
  }
}
```

Figure 38 - RDS Stack

After getting all the configurations, the project was deployed, and we could get all the connection information and cluster status from the AWS RDS interface represented in Figure 39. This information can be used to access the database using a MySQL client interface like *Datagrip*, *MySql Workbench* or *dBeaver* for example, and to configure the main service.

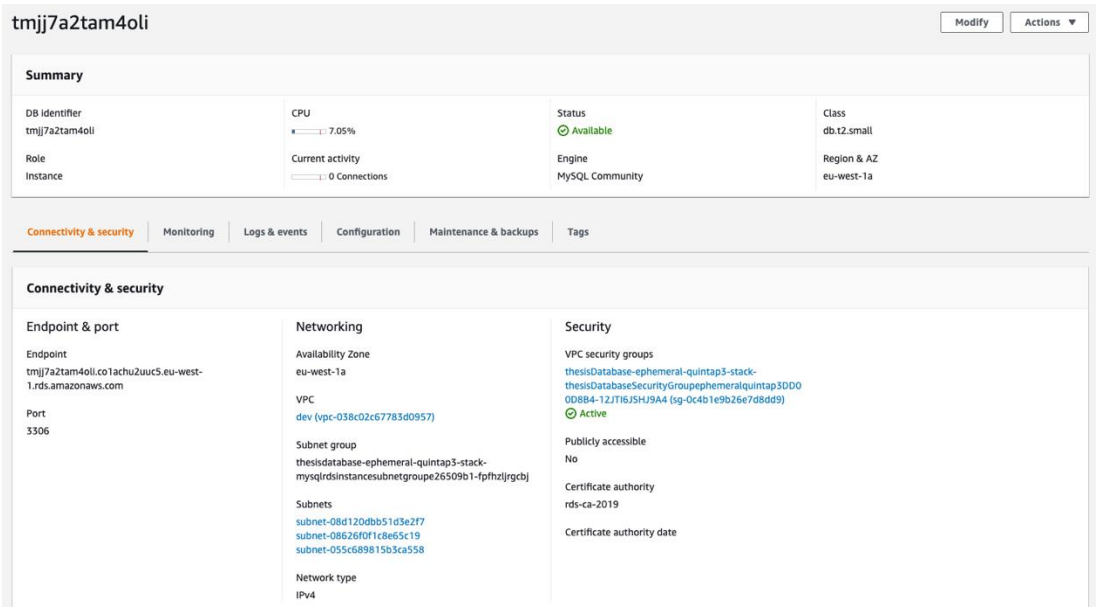


Figure 39 - AWS RDS Interface

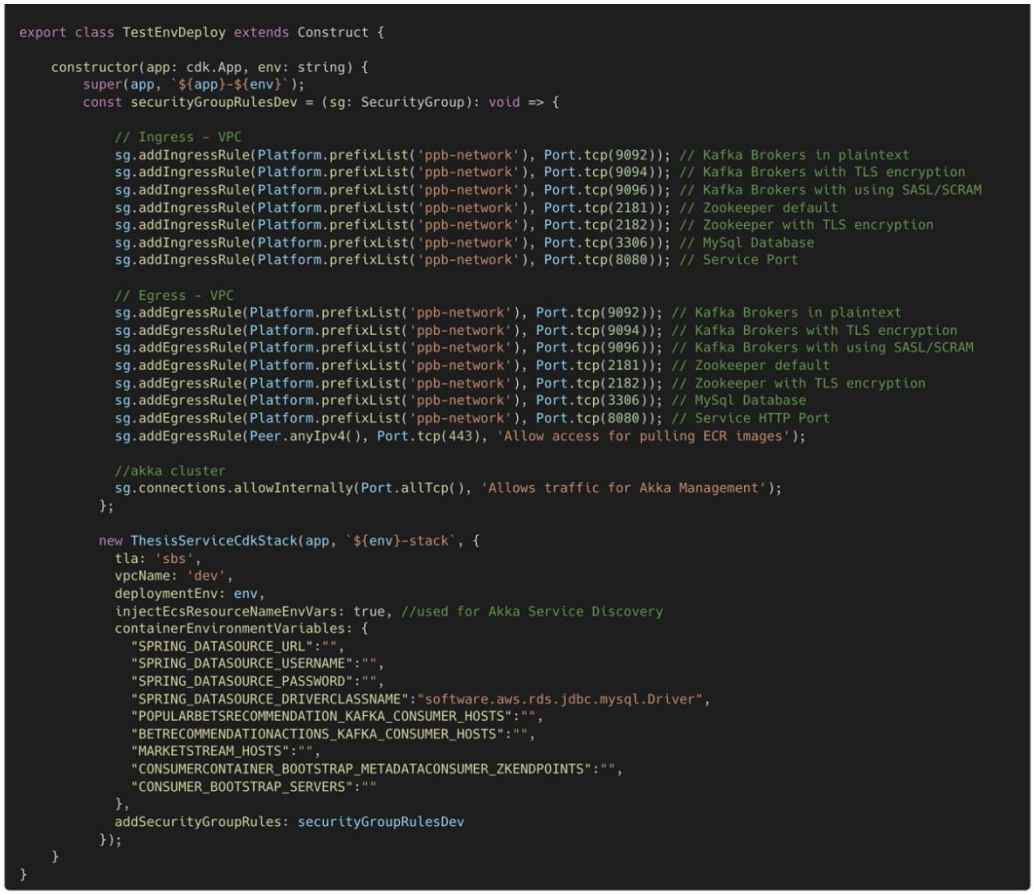
Now that all the services used by SBS are deployed, CDK code can be developed for it.

5.5.2.3 Spring Boot Application

For a proper functioning of the service in AWS using containers, some changes in the service and its configurations were necessary. All this work will be described later in the Service Changes chapter.

SBS CDK Infrastructure Code will have some generic code that could be applied to any ECS Cluster but there will be some specific methods for SBS. The main class and the environment file will be very similar to the structure used in the previous components, varying only in the ports opened for the Security Group. The code represented in Figure 40 demonstrates the test environment file created for the SBS and all the ports that were necessary to open for the component to work.

Since the service communicates with an Apache Kafka cluster and an RDS database, it was necessary to open ports to access these services and some specific to the Akka operation, which will be explained later. The service image was placed in an Amazon ECR repository for possible versioning and easy access by the deployment.



```
export class TestEnvDeploy extends Construct {

    constructor(app: cdk.App, env: string) {
        super(app, `${app}-${env}`);
        const securityGroupRulesDev = (sg: SecurityGroup): void => {

            // Ingress - VPC
            sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(9092)); // Kafka Brokers in plaintext
            sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(9094)); // Kafka Brokers with TLS encryption
            sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(9096)); // Kafka Brokers with using SASL/SCRAM
            sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(2181)); // Zookeeper default
            sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(2182)); // Zookeeper with TLS encryption
            sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(3306)); // MySql Database
            sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(8080)); // Service Port

            // Egress - VPC
            sg.addEgressRule(Platform.prefixList('ppb-network'), Port.tcp(9092)); // Kafka Brokers in plaintext
            sg.addEgressRule(Platform.prefixList('ppb-network'), Port.tcp(9094)); // Kafka Brokers with TLS encryption
            sg.addEgressRule(Platform.prefixList('ppb-network'), Port.tcp(9096)); // Kafka Brokers with using SASL/SCRAM
            sg.addEgressRule(Platform.prefixList('ppb-network'), Port.tcp(2181)); // Zookeeper default
            sg.addEgressRule(Platform.prefixList('ppb-network'), Port.tcp(2182)); // Zookeeper with TLS encryption
            sg.addEgressRule(Platform.prefixList('ppb-network'), Port.tcp(3306)); // MySql Database
            sg.addEgressRule(Platform.prefixList('ppb-network'), Port.tcp(8080)); // Service HTTP Port
            sg.addEgressRule(Peer.anyIpv4(), Port.tcp(443), 'Allow access for pulling ECR images');

            //akka cluster
            sg.connections.allowInternally(Port.allTcp(), 'Allows traffic for Akka Management');
          };

          new ThesisServiceCdkStack(app, `${env}-stack`, {
            tla: 'sbs',
            vpcName: 'dev',
            deploymentEnv: env,
            injectEcsResourceNameEnvVars: true, //used for Akka Service Discovery
            containerEnvironmentVariables: {
              "SPRING_DATASOURCE_URL":"",
              "SPRING_DATASOURCE_USERNAME":"",
              "SPRING_DATASOURCE_PASSWORD":"",
              "SPRING_DATASOURCE_DRIVERCLASSNAME":"software.aws.rds.jdbc.mysql.Driver",
              "POPULARBETSRECOMMENDATION_KAFKA_CONSUMER_HOSTS":"",
              "BETRECOMMENDATIONACTIONS_KAFKA_CONSUMER_HOSTS":"",
              "MARKETSTREAM_HOSTS":"",
              "CONSUMERCONTAINER_BOOTSTRAP_METADATACONSUMER_ZKENDPOINTS":"",
              "CONSUMER_BOOTSTRAP_SERVERS":""
            },
            addSecurityGroupRules: securityGroupRulesDev
          });
    }
}
```

Figure 40 - SBS Environment File with Security Group Rules

The main stack was developed using a structure that can be reused for other ECS clusters with the only difference being the addition of some service discovery logic, which is necessary for this service because all instances must know about each other. The stack is represented in Figure 41 and all the steps are numbered to have a simpler reading.

```
export class ThesisServiceCdkStack extends Stack {

  private readonly tla: string;
  private readonly vpc: ec2.IVpc;
  private readonly deploymentEnv: string;
  readonly cluster: CfnCluster;
  readonly logDriver: ecs.AwsLogDriver;

  constructor(scope: Construct, id: string, props: ThesisStackProps) {
    super(scope, id, props);

    this.tla = props.tla;

    //1. Getting Execution Policies for Task Role
    const executionRolePolicy = this.getPolicies();

    //2. Define VPC Settings
    //Using Company VPC for the specific environment
    //For this test, we will use dev env VPC only
    this.vpc = Platform.vpc(this, 'VPC', props.vpcName);
    this.deploymentEnv = props.deploymentEnv;

    //3. Define Log Driver
    //this.logDriver = this.getGrafanaLogging()
    this.logDriver = this.createAwsLogDriver(props);

    //4. Define Security Group
    const sg = this.getSecurityGroup();
    props.addSecurityGroupRules(sg);

    //5. Generate Cluster Name for Akka - Needed for Akka Service Discovery
    const clusterName = this.generateClusterName(this, props);

    //6. ECS Fargate Cluster for the Spring Boot Application
    const springbootEcsCluster = new ecs.Cluster(this, "springboot-ecs", {
      vpc: this.vpc,
      clusterName: clusterName
    })

    // Elastic Container Service Repository - to pull service image
    const ecrRepo = '737316999524.dkr.ecr.eu-west-1.amazonaws.com/quintap3springbootrepo';
    const version = 'latest'

    //7. Create Task Definition
    const taskDefinition = new ecs.FargateTaskDefinition(this, 'TaskDefinition');
    taskDefinition.addToExecutionRolePolicy(executionRolePolicy);
    taskDefinition.addToTaskRolePolicy(executionRolePolicy);

    //8. Create Service Name - Needed for Akka Service Discovery
    const serviceName = this.getServiceName(props);
    const containerEnvVars = this.getContainerVars(props, clusterName, serviceName)

    //9. Add container to the taskDefinition
    const container = taskDefinition.addContainer('container', {
      image: ecs.ContainerImage.fromRegistry(ecrRepo + ":" + version), //using an image from an ECR Repository
      environment: containerEnvVars,
      logging: this.logDriver,
      healthCheck: {
        command: [ "CMD-SHELL", "curl --request GET 'http://localhost:8080' || exit 1" ],
        startPeriod: cdk.Duration.seconds(300),
        timeout: cdk.Duration.seconds(60),
      },
    });

    container.addToExecutionPolicy(executionRolePolicy);
    container.addPortMappings({
      containerPort: 8080,
      protocol: ecs.Protocol.TCP,
    })

    //10. Application Deployment
    const springbootApp = new ecs_patterns.ApplicationLoadBalancedFargateService(this, 'springboot app svc', {
      cluster: springbootEcsCluster,
      serviceName: serviceName,
      desiredCount: 2, // Number of nodes
      cpu: 512, //CPU Information
      memoryLimitMiB: 1024, //Max Memory in MiB
      taskDefinition,
      securityGroups: [sg],
      publicLoadBalancer: false,
      assignPublicIp: true,
      taskSubnets: { subnetType: ec2.SubnetType.PRIVATE_WITH_NAT },
    })

    //11. Add execution Policies
    springbootApp.taskDefinition.addToExecutionRolePolicy(executionRolePolicy);
    springbootApp.taskDefinition.addToTaskRolePolicy(executionRolePolicy);
    springbootApp.loadBalancer.addSecurityGroup(sg);
  }
```

Figure 41 - SBS CDK ECS Stack

The stack consists of an ECS cluster on which a container of the SBS service is deployed. A Task Definition is created which defines some information regarding the service deployment and where the container and some of its configurations such as log driver, healthcheck and image will be specified. The container used will have specific environment variables to which two variables autogenerated by the CDK have been injected: ECS service name and ECS cluster name. These two variables will be useful for the service discovery that will be explored later.

After the deployment, it is possible to check the state of the service using the AWS ECS GUI and using the logs that are exported to AWS Cloudwatch as identified in the code in Figure 41.

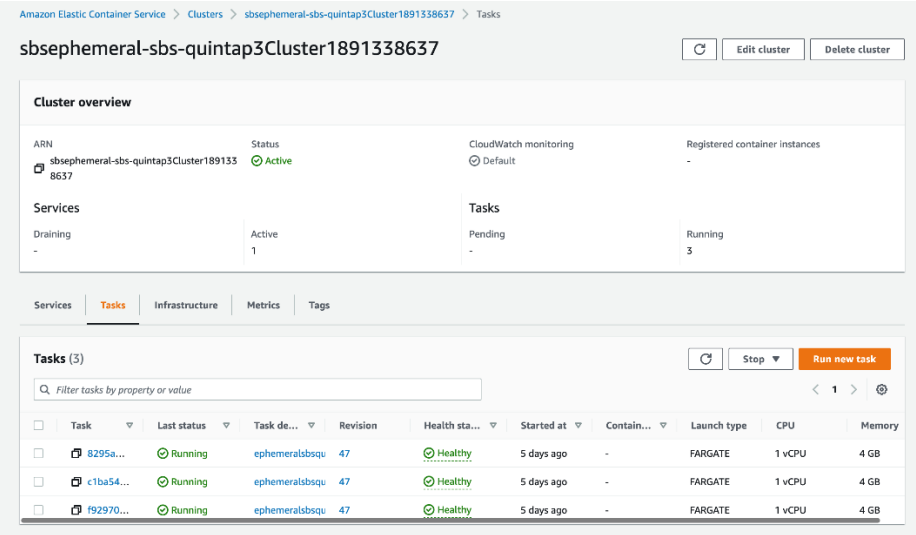


Figure 42 – ECS Tasks for SBS Service

As shown in the figure above, it is possible to see the number of active tasks of the service, their status, how long they have been running, among other information about the service. After selecting one of the tasks, you can also view the logs directly from this interface or open Cloudwatch for a better view of logs.

Now that all the infrastructure code has been analysed, it is also important to analyse and understand the changes that had to be made to the service for it to work with the new architecture and deployment type.

5.5.3 Service Changes

As mentioned during the state-of-the-art analysis, often services that are deployed in a private cloud may not be adapted for a direct migration to a public cloud. In this case, as a container-based deployment using an AWS ECS service was chosen, some changes were required for the SBS to work properly.

To understand the changes, it is first necessary to understand the service we are working with. SBS is a Spring Boot service developed in Kotlin and that uses Akka Actors for concurrency and

scalability. In the case of Akka, all instances must have knowledge of each other and usually when the service is started, a list of seed nodes is passed to all instances to know the hostname of each other.

For this service to work in containers, it took some research and some changes to the service and the CDK to make it all work. These changes are documented in more detail in Appendix G – Service Changes but the most important was the implementation of Akka Service Discovery that was one of the points necessary to explore and that led to some changes such as new lib dependencies for akka cluster management and akka discovery, removal of the seed nodes list and configuration of new service name and cluster name settings so that Akka could perform a search for nodes and identify them automatically. Akka in AWS is a subject still barely explored in the market and that led to an implementation from scratch in CDK to be able to inject all the necessary information in containers (Lightbend, n.d.).

A future improvement and implementation could be the use of AWS CloudMap which is a cloud resource discovery service. With Cloud Map, it is possible to define custom names for application resources, and it maintains the updated location of these dynamically changing resources. This increases the application availability because the web service always discovers the most up-to-date locations of its resources (AWS, n.d.).

5.5.4 Metrics, Logs and Alarmistic

Regarding the implementation of the Log Driver and how metrics and alarmistic will be collected for the service and components that have been deployed, a log driver was initially implemented for use by AWS CloudWatch. Amazon CloudWatch is a monitoring and observability service that provides data and actionable insights to monitor applications, respond to system-wide performance changes, and optimize resource utilization. CloudWatch collects monitoring and operational data in the form of logs, metrics, and events (*Amazon CloudWatch - Application and Infrastructure Monitoring*, n.d.).

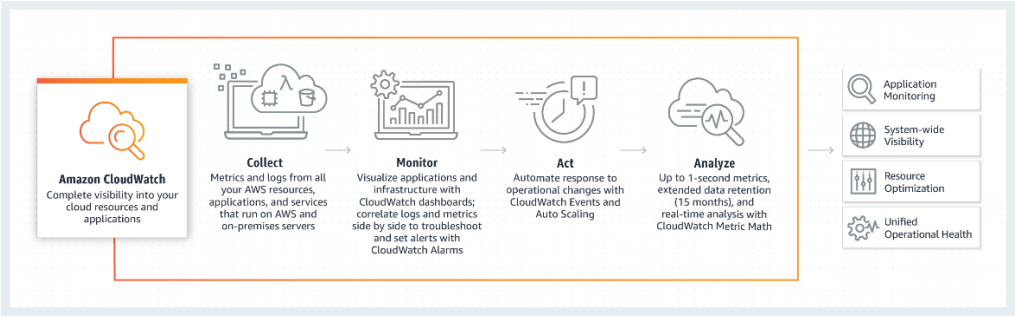


Figure 43 - AWS CloudWatch

Cloudwatch enables fast log analysis and monitoring using only the AWS interface but this also creates greater dependency on a single provider and an AWS monitoring implementation. For this reason, a log driver implementation has been added that enables the use of Grafana Cloud as a logging and monitoring service, represented in Figure 44.

```
private getGrafanaLogging() : LogDriver {

    return new FireLensLogDriver({
        options: {
            Name: 'grafana-loki',
            Url: 'https://<USER>:<PASS>@logs-dev-eu-west-0.grafana.net/loki/api/v1/push',
            Labels: '{job=\"firelens\"}',
            RemoveKeys: 'container_id,ecs_task_arn',
            LabelKeys: 'container_name,ecs_task_definition,source,ecs_cluster',
            LineFormat: 'key_value'
        }
    });
  }
```

Figure 44 - CDK Grafana Log Driver Implementation

5.5.5 Post Migration Analysis and Maintenance Work

After all the migration progress, the service will be tested by the development team to get the performance data and compare with the existent private cloud deployment. All this analysis will be made using some metrics that will be added by the team with a dashboard representation on Grafana Cloud. Some changes could be necessary depending on the test results, like changing CDK configurations to scale the number of tasks or changing some Akka properties.

For maintenance, developers should keep in mind that they must always release new images of the service when changes are made to it and these images must be versioned and sent to the ECR. For ease of deployment in environments, it is also possible to implement a pipelines system represented in the image below.

```
const tla = 'sbs';
const app = new cdk.App();

const ephemeralContextName = app.node.tryGetContext('ephemeral');

if (ephemeralContextName) {
    //Environment for testing
    new TestEnvDeploy(app, `ephemeral-${tla}`);
} else {

  const pipeline = new SinglePipelineStackV2(app, `${tla.toUpperCase()}-Pipeline-v1`, {
    tla,
    env: app.toolingEnvironment,
    githubConfigRepo: '<github repo here>',
    githubBranch: '<github branch here>',
    githubTokenSecretName: 'cicd/github-token',
    pipelineSuffix: '-v1',
  });

  pipeline.addDevApplicationStage(new NXTStage(app,`${tla.toUpperCase()}-Nxt-v1`, { env: app.devEnvironment }));

  pipeline.addPrdApplicationStage(new PRDStage(app,`${tla.toUpperCase()}-Prd-v1`, { env: app.prdEnvironment }));
}
```

Figure 45 - CDK Bin Stack Creation with Pipeline Implementation

The implementation represented in the Figure 45 gives the possibility to deploy an ephemeral just for testing and to deploy the service to every environment specified with a stage. To use ephemeral the user just needs to specify in the deploy command using *"--context ephemeral=username"*.

5.6 Migration Important Aspects and Conclusions

As expected, and previously mentioned, the migration was not direct and simple like in many other services that exist in private clouds. The migration process always requires an initially service analyse, outlining the steps to follow and the necessary changes.

The analysed service needed two dependencies (Apache Kafka and MySQL Database) which were also created in AWS taking into account the requirements, configurations and best practices for components of the same type.

This implementation demonstrated the need for documentation on both the service and the cloud, in this case AWS. The existence of documentation and CDK bases will enable a faster and more practical migration and deployment for development teams with less knowledge of AWS. For some issues raised during the migration, it is also important to have up-to-date documentation of the service to facilitate a quick response to problems or challenges encountered.

6 Experimentation and Evaluation

With the objective of developing a complete migration model between private cloud and public cloud, this model can be tested by taking as a source developed tests and feedback from users.

The main concept that will be subjected to an evaluation is the developed migration model, that will be compared against other existing models specified before in the Value Analysis.

6.1 Problem Description

Public cloud is increasingly being talked about and in recent years there has been a notable growth in the adoption of these services, not only for reasons of practicality but also of cost. For companies with several years in the market and with tens to thousands of different components under their infrastructure, it can be overwhelming and worrying sometimes to think about how to migrate each component without having high costs, not only in terms of time but also money and without affecting customers.

To make this process easier, it is necessary to have a model that not only assesses whether a component should be migrated but also identifies the steps that must be carried out from system specifications to component deployment and maintenance.

Removing this burden of long migrations and not knowing the steps to follow, you can not only save time and money, but also give developers greater freedom to make improvements and new features in the system.

6.2 Objectives

The main objective of this dissertation is to create an optimal migration model for services from on-premise to cloud infrastructure by exploring several existing models of service migration between the two clouds to be able to develop a more complete model that manages to have all the good points from pre to post migration. With this model, it is desired to facilitate the migration process and also, in some cases, to adapt and/or improve the components.

It is intended to test the model designed, applying it in the migration of services and later evaluate possible improvements of this model.

6.3 Hypotheses

Evaluation identifiers are intended to assist the evaluation of the project in question. Considering the objectives defined above, the hypotheses defined will be the following:

- H01 – The use of the designed model worsens the migration process between clouds
- H11 – The use of the designed model improves the migration process between clouds
- H02 – The model designed does not cover all the steps necessary for the migration
- H12 – The model designed tests the entire migration process
- H03 – The rate of user satisfaction is greater than 80%
- H13 – The rate of user satisfaction is less than 80%

To test these hypotheses, some quality surveys will be carried out to stakeholders and applying the *Wilcoxon* and *Shapiro-Wilk* method, it will be possible to answer the aforementioned hypotheses. The degree of user satisfaction must be greater than 80%, which is the approximate value defined by the *Customer Satisfaction Score* (CSAT) system (Grigore, 2022), which is a key indicator of user satisfaction. This indicates that user satisfaction is achieved when a degree equal to or greater than 80% is reached, so the higher the percentage, the greater the satisfaction achieved.

6.4 Identification of indicators and sources of information

The source of information will be the designed model and component migration tests following the model. This information, together with the responses to the stakeholder surveys, will provide data to answer the hypotheses.

6.5 Description of the evaluation methodology

After obtaining the answers to the questionnaires by the stakeholders, some statistical methods will be applied, such as the Shapiro-Wilk and Wilcoxon method, which will provide mathematical data for the analysis of the hypotheses.

6.6 Stakeholders Survey Data Analysis

As mentioned, a questionnaire was carried out to the stakeholders in which questions were asked to analyse the quality of the solution according to user satisfaction. The evaluation of the answers will be based on Hypothesis Tests in which there are two hypotheses H_0 and H_1:

- H_0 - The null hypothesis that is speculated to reject;
- H_1 - The alternative hypothesis to be verified.

To determine whether the null hypothesis should be rejected, the value of proof (*p-value*) is used, because it is considered the minimum value of significance that leads to the rejection of H_0. Thus, if *p-value* is less than or equal to the significance level of the test (α), H_0 is rejected. The significance level defined for the tests to be performed is 0.05, a value that corresponds to a confidence level of 95%.

6.6.1 Results Evaluation

This section assesses the results of the hypotheses defined, starting by identifying the questions formulated and the number of elements belonging to the sample of this analysis.

The questionnaire is composed of six questions which can be found in Appendix E – Stakeholders Survey and the sample of users is composed of eight people, all related to the development team that maintains the service. Each survey question will be evaluated independently, on the basis of hypothesis testing. Once the survey responses were obtained, tools such as Excel and RStudio were used as statistical computing softwares to perform the hypothesis tests. All tests performed in RStudio were performed in R language.

Taking into account the above information, the analysis performed for each question and the respective calculations are presented below.

Question 1 - Does the implemented solution enable a better understanding of the migration process to the public cloud?

As the sample for this evaluation is 8 elements, it is necessary to check whether these data follow a normal distribution. Therefore, the Shapiro-Wilk normality test is applied, taking into account the following hypotheses:

- H_0 - The sample values follow a normal distribution;
- H_1 - The sample values do not follow a normal distribution.

To apply the Shapiro-Wilk test, the R language can be used, which provides a method for testing a sample without having to calculate all the values of the Shapiro-Wilk formula or by applying the respective formula using Excel. Below we present the test performed in R, but the Excel calculations are available in Appendix F – Shapiro-Wilk Normality Test.

```
#QUESTION 1

q1 <- c(4,4,4,5,5,5,5,5)
shapiro.test(q1)

Shapiro-Wilk normality test

data:  q1
W = 0.6412, p-value = 0.0004791
```

Figure 46 - Shapiro-Wilk normality test for Question 1

In Figure 46, the Shapiro-Wilk Test is represented, in which the algorithm is applied to the vector of answers and where the p-value of 0.0004791 is obtained, which is lower than the significance level of 0.05 and which leads to the rejection of the null hypothesis, concluding with 95% confidence that the data of this question are not normally distributed.

Therefore, it is not possible to apply a parametric test to the sample, so a non-parametric test called the *Wilcoxon Test* will be applied.

Considering the hypothesis previously defined regarding user satisfaction, it is intended that this value be greater than 80%. Therefore, this value was converted to the scale used in the survey (scale 1-5) and concluded that satisfaction should be greater or equal than 4. Therefore, the following hypotheses were defined for the Wilcoxon Test:

- H_0 - The average of the responses obtained is less than or equal to 4;
- H_1 - The average of the responses obtained is greater than 4.

The application of Wilcoxon's Test is shown in Figure 47.

```
#QUESTION 1
q1 <- c(4,4,4,5,5,5,5,5)
wilcox.test(q1, mu=4, alternative="greater")

Wilcoxon signed rank test with continuity correction

data:  q1
V = 15, p-value = 0.01844
alternative hypothesis: true location is greater than 4
```

Figure 47 – Wilcoxon Test for Question 1

The analysis of Figure 46 leads to the conclusion that the obtained p-value was 0.01844. As this p-value is below the significance level, the hypothesis H_0 is rejected, allowing us to conclude, with a 95% confidence level, that the agreement of users with respect to the question under analysis is higher than 80%.

Now that question 1 has already passed the Shapiro-Wilk and Wilcoxon Tests analysis, we can proceed to the test of question 2.

Question 2 - Does the implemented solution facilitate the migration process?

In this question the following answer vector was obtained: "q2 <- c(5,5,5,5,5,5,5,5)". Since all users answered with the same value, it can be concluded that this is a discrete distribution with probability value 1. Thus, considering the hypothesis of satisfaction that we want to achieve (satisfaction level above 80%), the responses obtained allow us to conclude that user satisfaction on this question is above 80%.

Question 3 - Does the methodology cover the entire migration process?

In this question the following answer vector was obtained: "q3 <- c(4,4,5,5,5,5,5,5)". Given these values, we verify the normality of the distribution through the use of the Shapiro-Wilk test shown in Figure 48.

```
#QUESTION 3
q3 <- c(4,4,5,5,5,5,5,5)
shapiro.test(q3)

Shapiro-Wilk normality test

data:  q3
W = 0.56594, p-value = 0.00006323
```

Figure 48 - Shapiro-Wilk normality test for Question 3

In Figure 48, the Shapiro-Wilk test is represented, in which the algorithm is applied to the vector of answers and where the p-value of 0.00006323 is obtained, which is lower than the significance level of 0.05 and which leads to the rejection of the null hypothesis, concluding with 95% confidence that the data of this question are not normally distributed.

Therefore, the Wilcoxon Test will be applied to this data vector and is represented by the Figure 49.

```
#QUESTION 3
q3 <- c(4,4,5,5,5,5,5,5)
wilcox.test(q3, mu=4, alternative="greater")

Wilcoxon signed rank test with continuity correction

data:  q3
V = 21, p-value = 0.009828
alternative hypothesis: true location is greater than 4
```

Figure 49 - Wilcoxon Test for Question 3

The analysis of Figure 49 leads to the conclusion that the obtained p-value was 0.009828. As this p-value is below the significance level, the hypothesis H_0 is rejected, allowing us to conclude, with a 95% confidence level, that the agreement of users with respect to the question under analysis is higher than 80%.

Now that question 3 has already passed the Shapiro-Wilk and Wilcoxon Tests analysis, we can proceed to the test of question 4.

Question 4 - Is the maintenance process well defined?

The question 4 obtained the following answer vector: "q4 <- c(4,4,5,5,5,5,5)" and, for these values, we verify the normality of the distribution through the use of the Shapiro-Wilk test shown in Figure 50.

```
#QUESTION 4

q4 <- c(2,2,3,4,4,4,5,5)
shapiro.test(q4)

Shapiro-Wilk normality test

data:  q4
W = 0.87483, p-value = 0.1679
```

Figure 50 - Shapiro-Wilk normality test for Question 4

According to the results obtained, as the p-value is greater than the value of the significance level, there is no evidence to reject the null hypothesis, and the data are, at least approximately, from a normal distribution according to the Shapiro-Wilk test, with alpha significance level 0.05.

Question 5 - Do the base infrastructure models (CDK) for this methodology help in application development?

In this question the following answer vector was obtained: "q5 <- c(4,4,4,5,5,5,5,5)" as in question 1. Taking into account the results of the calculations performed for question 1, the hypothesis H_0 is rejected, allowing us to conclude, with a 95% confidence level, that the agreement of users with respect to the question under analysis is higher than 80%.

Question 6 - Taking into account the target public of the developed model, is the methodology fundamental in the migration of services?

In this question the following answer vector was obtained: "q6 <- c(5,5,5,5,5,5,5,5)" as in question 2 so, since all users answered with the same value, it can be concluded that this is a discrete distribution with probability value 1. Thus, considering the hypothesis of satisfaction that we want to achieve (satisfaction level above 80%), the responses obtained allow us to conclude that user satisfaction on this question is above 80%.

Taking into account the results of all the questions, customer satisfaction is clear but there is room for improvement in some parts of the process such as the post migration process.

7 Conclusion

This chapter presents the final conclusion of the dissertation, specifying the objectives achieved, the existing limitations and, finally, the future work that can be developed. This chapter concludes all the work carried out in this document, taking into account the conclusion of each of the chapters in order to achieve the objectives initially defined.

Starting with Chapter 1 that presents the problem and the approach that would be taken to solve it, addressing what research methodologies would be taken to achieve information on the topic. This chapter also defines the expected results and the structure in which the document is constructed.

Following the problem presentation, Chapter 2 creates a knowledge base that enables concepts to be understood ranging from cloud computing models to service models and the differences between the various types of cloud. An important point in this chapter is the research conducted on different models on the market, which allowed gathering ideas and understanding the customer and the points they were looking for to be fulfilled and that were a gap in the existing models.

Chapter 3 and 4 analyse different aspects but both influence the implementation documented in chapter 5, starting by comparing the desired model against existing models, followed by a problem analysis complemented by an initial solution design. The comparison of the models led to the conclusion that the most complete model and closest to the goal would be the model developed by Spotify (Carey, 2018) and, with the addition of some additional points, could be a good model to have as a reference. The framework that would be designed and developed by this document, would attempt to take the best points from the two best existing frameworks (Spotify Framework and The Framework for SMEs), reaching an optimal solution.

Chapter 4 looks more directly at the model to be developed from a design perspective, leading to an initial model represented by Figure 51 below. This model would cover not only the pre-migration steps but also the whole migration and post-migration maintenance process.

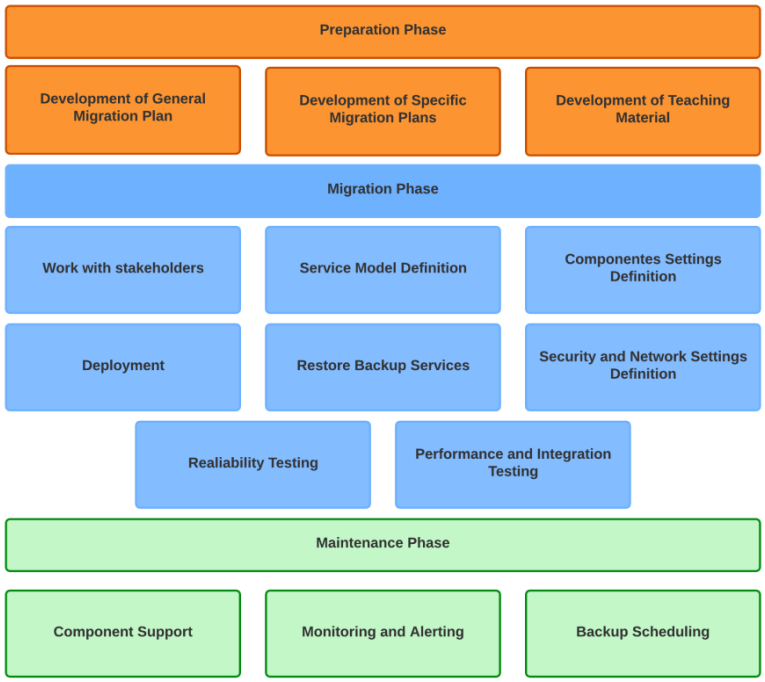


Figure 51 - Initial Model Design

Chapter 5 is the central chapter of the whole research, implementing the initial idea, migrating the service to the public cloud and concluding possible improvements and difficulties encountered. This whole process was initiated by an extensive study about the cloud and all the technologies and solutions presented by AWS, leading to an implementation of an ECS (Elastic Container Service) service using Docker Containers. All the implementation and code used are described in this chapter, addressing several difficulties and solutions created for them.

The entire investigation culminates in a final evaluation in Chapter 6, in which a survey is conducted to determine the quality of the solution, resulting in a high level of user satisfaction.

7.1 Achieved Objectives

This dissertation presents the various types of existing clouds and its main objective was to study the public cloud and the private cloud, outlining a plan/model for migrating services between clouds that would facilitate the work of teams on a daily basis.

This model was intended to migrate services from the private cloud to the public cloud, defining in this process not only the migration process but also the necessary steps before and after it.

The results were as expected, resulting in a model that helps in pre-migration planning including learning material, stakeholder meetings and specific migration plans, followed by a migration planning with the definition of the whole service deployment and configurations and, last but not least, the maintenance and support of the post-migration services.

The model was applied to a company service, in which all the CDK code was created from scratch and well organised so that it could be applied to other components in the future. The whole process was accompanied by meetings with the people responsible for the service and a careful analysis was carried out to understand how to adapt the service to the new cloud architecture. The final result was the one represented in Figure 52, obtaining a healthy service running on AWS without problems and with logs available for better debugging and analysis.

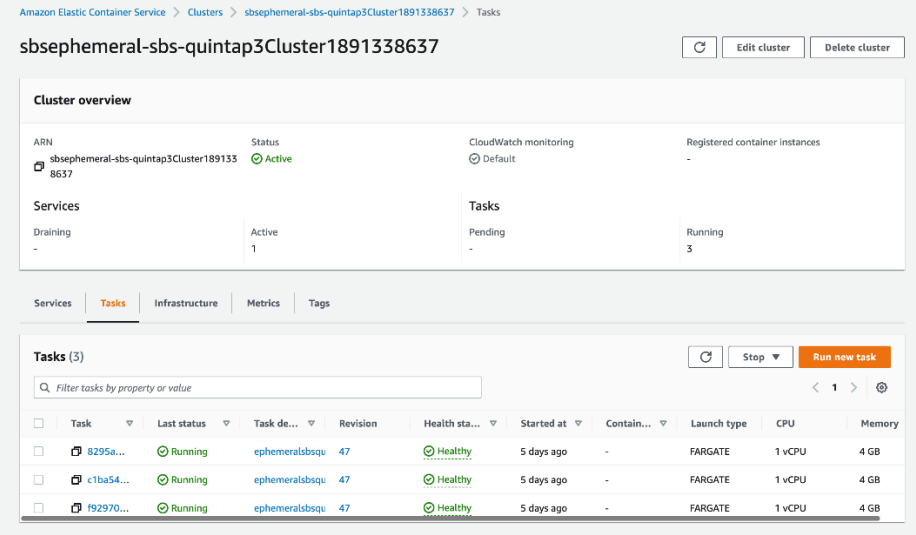


Figure 52 - Service Deployed in AWS

The pre-migration and migration process was well developed, with documentation on the process, cloud research and analysis of the service, achieving the objectives initially set. The post-migration process was covered briefly and has room for improvement and additions as mentioned later in the definition of the future work.

7.2 Constraints

This project had some limitations that slowed down the development process among them the adaptation of an Akka service to use containers and some adaptation work to use the AWS CDK. As mentioned in previous chapters, Akka in the public cloud with containers is a poorly documented topic and so it took some extra work to achieve a functional solution. With AWS CDK, for some of the services it required a better study such as the RDS database and also a search for the best ways to write and specify the required resources in CDK code.

7.3 Future Work

From the very beginning of the development, the subject of this dissertation showed to be a topic with a lot of room for research, improvement and development. After the development of the model, it is noticeable that it can still be improved and, as initially mentioned, to improve the process it will be necessary to develop base and example code for all types of applications, development of documentation, creation or use of a performance test tool to validate the application behaviour and the creation of a team to constantly support this type of work.

It would be an important step to create a repository that would extend the CDK base code from AWS and that would create examples of several types of deployment models enabling the deployment of ECS, EC2, EKS, among other solutions. This would not only create documentation and code rules for this type of infrastructure code but would also save a lot of time and money to the company making the process of creation or migration services much faster and simpler.

The migration would not be completed successfully without first validating and evaluating the components against the private cloud services. To do so, an integration and system test tool or framework could be created or implemented to test the service and also the entire flow between it and its dependencies.

As mentioned earlier, a process without a team to maintain and support it will never be a successful model. For this to have a significant impact and improvement in the work of a company, there must be a team to evolve it and support the entire process, providing guarantees of quality and efficiency.

References

ACM Digital Library. (n.d.). ACM Digital Library. Retrieved February 13, 2022, from https://dl.acm.org/

Alexander Osterwalder & Yves Pigneur. (2010). *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. John Wiley & Sons, Inc.

Amazon AWS. (n.d.). *What is the AWS CDK? - AWS Cloud Development Kit (CDK) v2*. AWS Documentation. Retrieved May 7, 2022, from https://docs.aws.amazon.com/cdk/v2/guide/home.html

Amazon CloudWatch—Application and Infrastructure Monitoring. (n.d.). Amazon Web Services, Inc. Retrieved August 27, 2022, from https://aws.amazon.com/cloudwatch/

AWS. (n.d.). *AWS Cloud Map Documentation*. Retrieved August 27, 2022, from https://docs.aws.amazon.com/cloud-map/?id=docs_gateway

AWS Fargate: What Are The Positives and Negatives? (n.d.). Opsani. Retrieved May 14, 2022, from https://opsani.com/resources/aws-fargate-what-are-the-positives-and-negatives/

Balobaid, A., & Debnath, D. (2020). An Effective Approach to Cloud Migration for Small and Medium Enterprises (SMEs). *2020 IEEE International Conference on Smart Cloud (SmartCloud)*, 7–12. https://doi.org/10.1109/SmartCloud49737.2020.00011

Bouayad, H., Benabbou, L., & Berrado, A. (2018). An Analytic Hierarchy Process based approach for Information technology governance framework selection. *Proceedings of the 12th International Conference on Intelligent Systems: Theories and Applications*, 1–6. https://doi.org/10.1145/3289402.3289515

Carey, S. (2018, July 30). *How Spotify migrated everything from on-premise to Google Cloud*. Computerworld. https://www.computerworld.com/article/3427799/how-spotify-migrated-everything-from-on-premise-to-google-cloud-platform.html

Google Cloud Tech (Director). (2018, July 26). *Spotify's Journey to the Cloud (Cloud Next '18)*. https://www.youtube.com/watch?v=5aBORQim-KM

Grafana: The open observability platform. (n.d.). Grafana Labs. Retrieved May 24, 2022, from https://grafana.com/

Grigore. (2022, April 7). *The Complete Guide to CSAT: Definition, Calculation & 2022 Benchmarks*. Retently. https://www.retently.com/blog/customer-satisfaction-score-csat/

James Bond. (2015). *The Enterprise Cloud – Best Practices for Transforming Legacy IT*. O'Reilly Media, Inc.

Jono Hey. (n.d.). *Conway's Law*. Sketchplanations. Retrieved January 14, 2022, from https://sketchplanations.vercel.app/conways-law

Judith Hurwitz, Marcia Kaufman, Dr. Fern Halper, & Daniel Kirsch. (2012). *Hybrid Cloud For Dummies*. John Wiley & Sons, Inc.

Kenneth B. Kahn. (2005). *The PDMA Handbook of New Product Development* (Second). John Wiley & Sons, Inc.

Khajeh-Hosseini, A., Greenwood, D., & Sommerville, I. (2010). Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS. *2010 IEEE 3rd International Conference on Cloud Computing*, 450–457. https://doi.org/10.1109/CLOUD.2010.37

Lightbend. (n.d.). *Akka Discovery Documentation*. Akka Documentation. Retrieved August 27, 2022, from https://doc.akka.io/docs/akka/current/discovery/index.html

Mansour, I. E. A., Bouchachia, H., & Cooper, K. (2017). Exploring Live Cloud Migration on Amazon EC2. *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, 366–371. https://doi.org/10.1109/FiCloud.2017.20

Mansour, I., Sahandi, R., Cooper, K., & Warman, A. (2016). Interoperability in the Heterogeneous Cloud Environment: A Survey of Recent User-centric Approaches. *Proceedings of the International Conference on Internet of Things and Cloud Computing*, 1–7. https://doi.org/10.1145/2896387.2896447

Mell, P., & Grance, T. (2011). *The NIST Definition of Cloud Computing*. 7.

Michael Kavis. (2014). *Architecting The Cloud – Design Decisions For Cloud Computing Service Models (SaaS, PaaS, And IaaS)*. John Wiley & Sons, Inc.

Morgan Perry. (2022, March 11). *Deploying Docker Containers on AWS: Elastic Beanstalk vs ECS vs EKS*. Qovery. https://www.qovery.com/blog/deploying-containers-on-aws-elastic-beanstalk-vs-ecs-vs-eks

Overview of Deployment Options on AWS - AWS Whitepaper. (n.d.). 24.

Pamami, P., Jain, A., & Sharma, N. (2019). Cloud Migration Metamodel: A framework for legacy to cloud migration. *2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, 43–50. https://doi.org/10.1109/CONFLUENCE.2019.8776983

Quality-One International. (2017, December 19). *Quality Function Deployment*. https://quality-one.com/qfd/

Spotify Case Study. (n.d.). Google Cloud. Retrieved January 27, 2022, from https://cloud.google.com/customers/spotify

Synergy Research Group. (2021, October 28). *Amazon, Microsoft & Google Grab the Big Numbers – But Rest of Cloud Market Still Grows by 27% | Synergy Research Group*.

Synergy Research. https://www.srgresearch.com/articles/amazon-microsoft-google-grab-the-big-numbers-but-rest-of-cloud-market-still-grows-by-27

What is Amazon EC2? - Amazon Elastic Compute Cloud. (n.d.). AWS Documentation. Retrieved May 14, 2022, from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

Appendixes

Appendix A – Cloud Study

1. Public and Private Cloud Use from SMEs to Big Companies

Having the definition of cloud and its different models clear, it can be addressed the topic of using Cloud Services in companies, ranging from the smallest and medium-sized companies to large companies such as Google, Facebook or Spotify.
First by talking about the importance and impact of choosing the model and type of cloud to use, the different points of analysis when choosing one model and the main challenges of migrating between clouds will be addressed.

a. Cloud Use from Small and Medium (SME) businesses to Big Companies

In the current market, most companies have begun incorporating cloud-based technology into their infrastructure. According to a study by the Synergy Research Group, in the third quarter of 2021 around 45 billion dollars were spent in cloud infrastructure systems that is 37% higher than the spendings in the same quarter of 2020(Synergy Research Group, 2021). These values shows that the market is looking more and more for solutions in the Cloud Computing Services because these services offer good value for money, allow for greater scalability compared to their own infrastructure, and enable greater focus on the product and on delivering better quality to the customer.

When talking about Small and Medium Businesses, they were most likely to lean toward public clouds, with 44% of them choosing either single or multiple public clouds. This can be explained by the fact that they are smaller teams/companies that are unable to have a team specialized in infrastructure management and their own infrastructure due to its inherent costs.

Despite all of this, 98% of businesses still rely on on-premises infrastructures, according to a survey by Spiceworks of over 500 IT decision makers (Leon Yen, 2021). This survey also addresses the reason why companies buy new hardware every year, constantly expanding or upgrading their servers, usually related to business growth or server-related issues such as: performance degradation, reliability issues, hardware failures and maintenance costs. On Big Companies, this choice may vary depending on the company objectives, security concerns and having or not a team to maintain all the necessary servers.

b. Organizational Impact of the Cloud Model

Michal T. Kanazawa once said “People don’t hate change, they hate the way you’re trying to change them.”(Michael Kavis, 2014). Change is hard and people sometimes hate that because there is no clear path or model that can make everyone satisfied and that can clear all the possible problems that can be inherent to the change.

Choosing the Cloud Model (Service and/or Deployment) is not any different from any important change in the world of IT. When approaching the Deployment Model, this will be applied to all the company and need to be careful analysed before any decision. The Service Model pick can be more directly associated with the service/component that will be created or migrated.

At the end of the day, it all comes down to architecture. First, it is necessary to understand the business requirements and analyse the existing infrastructure and components so that it is possible to map the right cloud deployment and service models to the company needs (Michael Kavis, 2014).

All this changes not only affect the IT department but also the rest of the company as it leads to a change in the business operating model. Before Cloud Computing until today, many companies chose to have their own servers over which they had full control and which they customized to their liking and need, using a model called on-premise software delivery model or enterprise model. This model over time has demonstrated certain needs that have been filled in the market by cloud computing and a new model called Elastic Cloud model (Michael Kavis, 2014). While in the first model, the company controls the entire infrastructure and its components, in the elastic model, the company controls only the components, leaving control of the infrastructure to an external provider. Both models impact the company in several areas that not only IT, and when changing from an enterprise model to an elastic model there are some topics that must be taken into account due to their impact:

- Deployment - Elastic model has the great advantage of having frequently deployments and presenting zero downtime compared to the lower number of releases in the on-premise model that often depended on technicians and sometimes impacted by the infrastructure itself.
- Monitoring – Having an infrastructure that adapts to the number of requests and to the needs in real time also carries the responsibility of having a greater concern and monitoring of the system to have an idea of the cost, the state of the service and the functioning over time.
- Availability - In on-premise systems, it is often difficult to guarantee 100% availability as problems caused by the infrastructure often impact components affecting customer service. In a more elastic and reactive system, certain SLAs are defined between the provider and the customer, the provider having to guarantee that the system will be available within the values defined in the contract. On the other hand,

the company must guarantee a quality application, which can be updated easily with constant deploys and that can scale to support increased or decreased traffic.

- Customer Support - Having an elastic system, the responsibility of the support of the infrastructure would be moved to the cloud provider and the company having the sole responsibility to support, in some cases, the application part depending on the chosen service model.
- Finance - Changing model also influences a lot the company's Accounting and Financing. Being that in the on-premise software delivery model we have a predictable initial expense, with an investment in servers and software, even before we have developed components. From the company perspective, the initial investment in this model is large and impacts in a negative way their cash flow. On the other hand, in the elastic model, a pay-as-you-go contract is often agreed between the customer and the provisioning company, which favours the company in its initial investment, which should be much lower than in the previous model. Instead of paying a large amount of money in the beginning, the company can pay in a rate that is proportional to the rate at which it brings in revenue or value to the organization or they can pay charges like a monthly fee. The elastic model has not only positive points because, in models such as pay-as-you-go, it is difficult to predict costs and profits as usage may vary over time. In the enterprise model, the company pay an initial cost, which is a one-time fixed cost and has an annual maintenance cost that normally are very predictable. In case of necessity to expand the servers and buy more resources, the company go through a procurement process, which is easily tracked (Michael Kavis, 2014).
- Legal – In legal matters, when joining cloud services, the process can be lengthy because there is a greater demand and concern in terms of security, privacy and policies. Depending on the applications and the business in which the company is included, the criteria and specifications related to security and privacy may vary according to laws, regulations and policies directly related to the business or country for which the application is intended. A best practice is to produce a document that spells out all of the policies and procedures related to privacy, security, regulations, SLAs, ownership, and so forth (Michael Kavis, 2014).
- Sales – If the company pretends to develop or deploy a developed component that will be used and sold to external clients, the salespeople must understand the basics of cloud computing and how everything work and what is the difference to the on-premise model. It is essential that the sales team knows how to explain points related to security and SLAs, as these are the points in which there is greater concern and attention from customers.
- Human Resources – Often when moving to a public cloud, companies will not have the people with the skills or knowledge needed to migrate and maintain components in this different type of computing. To tackle this problem, human resources will have the challenge of finding workers prepared for this challenging task. This process can be complicated as there is often a shortage of people with cloud knowledge, which can lead to the need to find people willing to relocate or

even hire remote workers or subcontractors to help with this process(Michael Kavis, 2014). The HR and IT teams should come together and carry out brainstorms that allow them to reach an optimal solution for everyone and that makes the best use of reward and recognition methods, and optimize the organization's structure to support the change that this migration will bring. All of these impacts and considerations can be illustrated by the Conway's Law that says "If the architecture of the system and the architecture of the organization are at odds, the architecture of the organization wins"(Justin Garrison & Kris Nova, 2018). If the company wants the change from on-premise model to elastic or hybrid model to be the most clear and smooth as possible, there must be an overall organization change and planning so that all changes and effects that this shift can affect on your product are considered.

c. Cloud Migration Challenges

When a company already has a large infrastructure, sometimes the decision to move from an on-premise cloud to something completely public or hybrid can be tricky for several reasons:

- Having a large datacenter that they already maintain: the company would need to decide what to do with that, depending on the chosen solution. Security and Privacy-related Concerns: 84% of IT professionals were worried about cloud security during the 2020 COVID-19 pandemic according to a survey from Fugue Company (“How Many Companies Use Cloud Computing and What’s It All About?,” 2021). These concerns are related to Cloud Infrastructure Bad Configurations, Unauthorized Accesses, Insecure APIs, Traffic Hijacking and external data sharing (Ivey, 2021).
- The cost of migrating or adapting components to public cloud.
- In case of using a hybrid architecture, they need to adapt components and frameworks related to the migration and movement of components between different clouds in case of need.
- Change a component to a true cloud application can be difficult and can cause changes in various architectural components
- The idea of loss of control of software/hardware assets.

This change can be hard as explained above but it is also driven by some negative points of on-premise use such as the cost of maintaining all components such as network devices, bare metal servers, software packages and the headache when it is necessary to scale the infrastructure to achieve certain goals or need and that often leads to a significant cost of planning, organization and deployment that would be avoided in the cloud.

1. Changing Component from Legacy Code to Cloud

As previously mentioned, changing a component from legacy to the cloud can be challenging because legacy systems are reliant on ACID principles unlike cloud systems which are based on BASE principles and focused on scalability.

ACID is composed by four principles: atomicity, consistency, isolation and durability. The main focus of these principles is the consistency before and after every transaction, that is a representation of a single logical unit of work which accesses and possibly modifies the contents of data. In ACID based components, a transaction is not complete until it is committed and the data is up to date, forcing data consistency.

In the cloud, the architecture is quite different and before moving a component there is the necessity to understand the changes and the architecture differences that are necessary to have a strong and stable component in the cloud, capable of scaling and adapting depending on the number of interactions with users. To achieve all this, Cloud Architecture rely on Basically Available, Soft State, Eventually Consistent (BASE) transactions, in other words, it ensures that in case resources fail, the information will eventually become consistent. BASE is often used in volatile environments where nodes may fail or systems need to work whether the user is connected to a network or not (Michael Kavis, 2014).

Architecting solutions for cloud computing requires a solid understanding of how the cloud works and every solution must be designed with the expectation that everything can and will fail. Most legacy systems are design to run in a single partition and expect the data to be always consistent, making many of these systems not directly compatible with the cloud-based architecture that requires partition tolerance. These systems were never intended to be built in a way that the system could automatically scale up or down as the number of transactions change. Traditionally, scaling techniques rely on vertical scaling, by increasing the number of CPUs, memory or disk space used by the component or replacing the existing infrastructure with a more powerful or upgraded hardware(Michael Kavis, 2014).

Partition tolerance means if one instance of a compute resource cannot complete the task, another instance is called on to finish the job. This architecture predicts that eventually the discrepancies will be reconciled and the data will be up to date (Michael Kavis, 2014).

Migrating single-partition applications to the cloud makes the migration act more like a hosting solution rather than a scalable cloud solution, not taking advantage of all the features that the cloud provides and, sometimes, negatively affecting the performance of the system and the quality of the service provided to the user.

d. IT Starts with Architecture

Often in the IT world, teams try to rush quickly into the development part without having a clear definition of functional and non-functional criteria. This constant running and jumping the planning process can have higher costs in the future and can lead to bad decisions in the creation and development of components. As Frank Lloyd, an American architect, said "A doctor can bury his mistakes but an architect can only advise his clients to plant vines" (Michael Kavis, 2014) and in the development of components for the Cloud this is even more important as the risk is greater when shifting more control to cloud providers.

When we try to find a solution to a problem, be it the choice of a deployment model or the best way to develop an application, there is a very simple but valuable rule that helps in these

decisions: “Why? Who? What? Where? When? How?". First, we need to understand the problem we are trying to solve and **why** our business needs that. Second, we need to point **who** need this problem solved and **what** are the requirements for this. Next, we must identify **where** this service will be used and identify restrictions whether from country regulations, language or other usability concerns. After all this, we must identify costs and **when** this service is needed, so that it is possible to identify **how** the company will develop the service.

e. Cloud Computing Worst Practices

In this subchapter, some practices or ideas that often lead to a misuse of the cloud will be addressed.

1. Architecture

Most legacy architectures were never intended to be built in a manner where the system automatically scales as the number of transactions increases as we normally see in the cloud-based components.

Typically, component scaling techniques rely heavily on vertical scaling, that is, increasing hardware such as CPU, memory or disk. This type of scaling is called scaling up, and it only requires hardware changes, not requiring any component changes.

It is often difficult to migrate applications to the cloud because a lot of software is written in order to take full advantage of the infrastructure, which makes it very coupled to it, making it difficult to move to an elastic component architecture deployed in the cloud. If elasticity is not the main reason for moving to the cloud, then what the company most likely need is a hosting solution, that is not the same as cloud computing. “Hosting does not provide for the five characteristics of cloud computing: broad network access, elasticity, measured service, on-demand self-service, and resource pooling. Hosting is simply renting or buying infrastructure and floor space at a hosting provider’s facility”(Michael Kavis, 2014).

Another challenge for legacy applications is whether the system design is stateful or stateless. Cloud services are stateless, that is a service that is unaware of any information from previous request and is only aware of information while the service is processing a given request. Being that a stateless service stores the application state on the client and not in the server, so there is no dependency on the infrastructure. In case we have an on-premise service that has been designed and architected to be a stateful service, then migrating to the cloud either will take a major reengineering effort, will not reap many of the benefits of the cloud, or might be completely unfeasible. Because of this, companies that migrate legacy stateful applications to the cloud will likely be disappointed with the end result if they expect to reap all of the benefits of cloud computing.

The best decision is, first of all, to make sure that a clear analysis of the existing components is made and that there is a clear understanding by everyone on the difference between stateless and stateful.

2. Expectations

One of the biggest misguided perceptions of cloud computing is that cloud will greatly reduce the cost of doing business. That may be true for some initiatives but not all of them. The cost is directly related to how the company is able to make good use of the contracted resources, effectively optimizing the architecture of its components. For this, the use of the services and their price over time must be monitored.

Even with a monitoring effort over time, not all cases are possible to be solved by cloud computing so it is necessary to take this into account.

3. Misinformation

It is necessary to make an initial analysis that allows to have a realistic expectation of the cost and use of the component so that it is known if it is worth the migration or not. An initiative should be broken down into small deliveries, so that it is possible to deliver value more quickly and teams can learn over time. Trying to migrate everything at once will only bring problems and more costs than necessary.

The best way to optimize the use of resources will be the constant monitoring and optimization of the components, taking into account the consumption and cost of the component at that moment.

Another myth that is often talked about cloud computing is that it poses greater security risks than on-premises cloud, which is not true. With the proper security architecture, the public cloud can be more secure than most datacenters. Unfortunately, very few corporations know enough about security in the cloud to be able to architect for it, and many others do not have the skill set internally to build the appropriate level of security.

4. Select the Favourite instead of the Appropriate

A very common mistake is choosing the provider because the company is more familiar with it or because it is the most famous. First, an analysis must be made of what components will be like to migrate and there must be a clear definition of the three types of cloud service (SaaS, IaaS and PaaS). Keeping this in mind, it will be easier to know which type is suitable for each component and also to analyse in the market which provider will be more favourable and appropriate, not only in monetary terms but in terms of feasibility, performance, security and support.

5. Out of Business Scenarios

When signing a contract with a cloud service, we must always keep in mind that everything can and will fail, nothing is 100% stable and the cloud provider's servers are basically normal servers that can have problems. All services must bear this in mind and be designed and prepared for failure.

A good practice is to understand the cloud provider's SLAs, understand the contract policies and review all legal agreements. With all this data, the company must analyse

whether the best solution is to have everything in the cloud or not, depending on the risk it can take.

6. Costs

One of the promises of cloud computing is that the pay-as-you-go model will save the company money because they can pay according to what they need, not having a heavy and fixed cost monthly or annually. This only holds true if the software is architected and managed in a way that optimizes the use of cloud services. One of the most significant aspects of cloud computing is the speed with which services or computing resources may be turned on. However, if the process of consuming cloud resources is not tightly regulated, monthly expenses can increase.

To prevent these unexpected costs, you must be clear about the cost of each cloud service model and control each service to optimize it and control and monitor costs. Don't underestimate the work necessary to integrate with old architectures, as well as the expenses of educating existing personnel or acquiring skilled engineers, for firms with legacy systems.

Appendix B – AHP Analysis

Comparações par a par

	DiffServModel	GenerPlan	SpecPlan	PreMigSt	ValidStake	LiftEasy	RewriteCp	PostMigVal
DiffServModels	1,00	0,14	0,20	0,14	0,20	0,33	0,33	0,20
GenerPlan	7,00	1,00	5,00	3,00	3,00	3,00	5,00	3,00
SpecPlan	5,00	0,20	1,00	0,33	0,33	3,00	3,00	0,33
PreMigSt	7,00	0,33	3,00	1,00	3,00	3,00	3,00	3,00
ValidStake	5,00	0,33	3,00	0,33	1,00	3,00	3,00	0,33
LiftEasy	3,00	0,33	0,33	0,33	0,33	1,00	3,00	0,33
RewriteCp	3,00	0,20	0,33	0,33	0,33	0,33	1,00	0,33
PostMigVal	5,00	0,33	3,00	0,33	3,00	3,00	3,00	1,00
SUM	36,00	2,88	15,87	5,81	11,20	16,67	21,33	8,53

Matriz de comparação normalizada e pesos estimados

	DiffServModel	GenerPlan	SpecPlan	PreMigSt	ValidStake	LiftEasy	RewriteCp	PostMigVal	Weights
DiffServModels	0,03	0,05	0,01	0,02	0,02	0,02	0,02	0,02	0,023945188
GenerPlan	0,19	0,35	0,32	0,52	0,27	0,18	0,23	0,35	0,300930087
SpecPlan	0,14	0,07	0,06	0,06	0,03	0,18	0,14	0,04	0,089784622
PreMigSt	0,19	0,12	0,19	0,17	0,27	0,18	0,14	0,35	0,201448738
ValidStake	0,14	0,12	0,19	0,06	0,09	0,18	0,14	0,04	0,118776103
LiftEasy	0,08	0,12	0,02	0,06	0,03	0,06	0,14	0,04	0,068382779
RewriteCp	0,08	0,07	0,02	0,06	0,03	0,02	0,05	0,04	0,045869327
PostMigVal	0,14	0,12	0,19	0,06	0,27	0,18	0,14	0,12	0,150863156
SUM	1	1	1	1	1	1	1	1	

										CriteriaWeights		Weighted Sum Value
1,00	0,14	0,20	0,14	0,20	0,33	0,33	0,20	x	0,023945	=	0,21
7,00	1,00	5,00	3,00	3,00	3,00	5,00	3,00	x	0,300930	=	2,77
5,00	0,20	1,00	0,33	0,33	3,00	3,00	0,33	x	0,089785	=	0,77
7,00	0,33	3,00	1,00	3,00	3,00	3,00	3,00	x	0,201449	=	1,89
5,00	0,33	3,00	0,33	1,00	3,00	3,00	0,33	x	0,118776	=	1,07
3,00	0,33	0,33	0,33	0,33	1,00	3,00	0,33	x	0,068383	=	0,57
3,00	0,20	0,33	0,33	0,33	0,33	1,00	0,33	x	0,045869	=	0,39
5,00	0,33	3,00	0,33	3,00	3,00	3,00	1,00	x	0,150863	=	1,41

Random Index (RI)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

Weighted Sum Value	STEP2	STEP3	CI STEP4	CR=CI/RI STEP5
0,21	8,590	8,845673	0,12081	0,085681
2,77	9,189			
0,77	8,570			
1,89	9,384			
1,07	8,995			
0,57	8,264			
0,39	8,451			
1,41	9,323			

Matrizes com alternativas

DiffServModels		PropMig	SMEfram	CloudModel	CloudToolkit	LivCloud	SpotifyMod
	PropMig	1,00	3,00	3,00	5,00	9,00	3,00
	SMEfram	0,33	1,00	3,00	5,00	5,00	0,33
	CloudModel	0,33	0,33	1,00	5,00	3,00	0,20
	CloudToolkit	0,20	0,20	0,20	1,00	3,00	0,14
	LivCloud	0,11	0,20	0,33	0,33	1,00	0,11
	SpotifyMod	0,33	3,00	5,00	7,00	9,00	1,00
	SUM	2,31	7,73	12,53	23,33	30,00	4,79

GenerPlan		PropMig	SMEfram	CloudModel	CloudToolkit	LivCloud	SpotifyMod
	PropMig	1,00	3,00	3,00	5,00	7,00	3,00
	SMEfram	0,33	1,00	5,00	5,00	7,00	0,33
	CloudModel	0,33	0,20	1,00	3,00	3,00	0,33
	CloudToolkit	0,20	0,20	0,33	1,00	3,00	0,20
	LivCloud	0,14	0,14	0,33	0,33	1,00	0,14
	SpotifyMod	0,33	3,00	3,00	5,00	7,00	1,00
	SUM	2,34	7,54	12,67	19,33	28,00	5,01

SpecPlan		PropMig	SMEfram	CloudModel	CloudToolkit	LivCloud	SpotifyMod
	PropMig	1,00	3,00	3,00	7,00	5,00	2,00
	SMEfram	0,33	1,00	3,00	5,00	5,00	0,33
	CloudModel	0,33	0,33	1,00	5,00	5,00	0,33
	CloudToolkit	0,14	0,20	0,20	1,00	3,00	0,20
	LivCloud	0,20	0,20	0,20	0,33	1,00	0,20
	SpotifyMod	0,50	3,00	3,00	5,00	5,00	1,00
	SUM	2,51	7,73	10,40	23,33	24,00	4,07

PreMigSt		PropMig	SMEfram	CloudModel	CloudToolkit	LivCloud	SpotifyMod
	PropMig	1,00	2,00	2,00	3,00	5,00	1,00
	SMEfram	0,50	1,00	3,00	5,00	5,00	0,33
	CloudModel	0,50	0,33	1,00	3,00	5,00	0,33
	CloudToolkit	0,33	0,20	0,33	1,00	3,00	0,20
	LivCloud	0,20	0,20	0,20	0,33	1,00	0,14
	SpotifyMod	1,00	3,00	3,00	5,00	7,00	1,00
	SUM	3,53	6,73	9,53	17,33	26,00	3,01

ValidStake		PropMig	SMEfram	CloudModel	CloudToolkit	LivCloud	SpotifyMod
	PropMig	1,00	5,00	5,00	2,00	7,00	2,00
	SMEfram	0,20	1,00	3,00	0,33	5,00	0,20
	CloudModel	0,20	0,33	1,00	0,20	5,00	0,20
	CloudToolkit	0,50	3,00	5,00	1,00	7,00	0,33
	LivCloud	0,14	0,20	0,20	0,14	1,00	0,14
	SpotifyMod	0,50	5,00	5,00	3,00	7,00	1,00
	SUM	2,54	14,53	19,20	6,68	32,00	3,88

LiftEasy		PropMig	SMEfram	CloudModel	CloudToolkit	LivCloud	SpotifyMod
	PropMig	1,00	2,00	3,00	3,00	3,00	1,00
	SMEfram	0,50	1,00	3,00	3,00	5,00	0,50
	CloudModel	0,33	0,33	1,00	3,00	5,00	0,33
	CloudToolkit	0,33	0,33	0,33	1,00	0,33	0,20
	LivCloud	0,33	0,20	0,20	3,00	1,00	0,20
	SpotifyMod	1,00	2,00	3,00	5,00	5,00	1,00
	SUM	3,50	5,87	10,53	18,00	19,33	3,23

RewriteCp		PropMig	SMEfram	CloudModel	CloudToolkit	LivCloud	SpotifyMod
	PropMig	1,00	5,00	3,00	7,00	7,00	1,00
	SMEfram	0,20	1,00	0,33	5,00	3,00	0,20
	CloudModel	0,33	3,00	1,00	5,00	7,00	0,33
	CloudToolkit	0,14	0,20	0,20	1,00	3,00	0,20
	LivCloud	0,14	0,33	0,14	0,33	1,00	0,20
	SpotifyMod	1,00	5,00	3,00	5,00	5,00	1,00
	SUM	2,82	14,53	7,68	23,33	26,00	2,93

PostMigVal		PropMig	SMEfram	CloudModel	CloudToolkit	LivCloud	SpotifyMod
	PropMig	1,00	2,00	5,00	5,00	7,00	1,00
	SMEfram	0,50	1,00	5,00	5,00	5,00	0,33
	CloudModel	0,20	0,20	1,00	1,00	3,00	0,20
	CloudToolkit	0,20	0,20	1,00	1,00	3,00	0,20
	LivCloud	0,14	0,20	0,33	0,33	1,00	0,20
	SpotifyMod	1,00	3,00	5,00	5,00	5,00	1,00
	SUM	3,04	6,60	17,33	17,33	24,00	2,93

Matrizes com alternativas normalizadas

DiffServModels		PropMig	SMEfram	CloudMod	CloudToo	LivCloud	SpotifyMod	Weights
	PropMig	0,4327	0,3879	0,2394	0,2143	0,3000	0,6267	0,3668
	SMEfram	0,1442	0,1293	0,2394	0,2143	0,1667	0,0696	0,1606
	CloudModel	0,1442	0,0431	0,0798	0,2143	0,1000	0,0418	0,1039
	CloudToolkit	0,0865	0,0259	0,0160	0,0429	0,1000	0,0298	0,0502
	LivCloud	0,0481	0,0259	0,0266	0,0143	0,0333	0,0232	0,0286
	SpotifyMod	0,1442	0,3879	0,3989	0,3000	0,3000	0,2089	0,2900
	SUM	1	1	1	1	1	1	1,00

GenerPlan		PropMig	SMEfram	CloudMod	CloudToo	LivCloud	SpotifyMod	Weights
	PropMig	0,4268	0,3977	0,2368	0,2586	0,2500	0,5989	0,3615
	SMEfram	0,1423	0,1326	0,3947	0,2586	0,2500	0,0665	0,2075
	CloudModel	0,1423	0,0265	0,0789	0,1552	0,1071	0,0665	0,0961
	CloudToolkit	0,0854	0,0265	0,0263	0,0517	0,1071	0,0399	0,0562
	LivCloud	0,0610	0,0189	0,0263	0,0172	0,0357	0,0285	0,0313
	SpotifyMod	0,1423	0,3977	0,2368	0,2586	0,2500	0,1996	0,2475
	SUM	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000

SpecPlan		PropMig	SMEfram	CloudMod	CloudToo	LivCloud	SpotifyMod	Weights
	PropMig	0,3985	0,3879	0,2885	0,3000	0,2083	0,4918	0,3458
	SMEfram	0,1328	0,1293	0,2885	0,2143	0,2083	0,0820	0,1759
	CloudModel	0,1328	0,0431	0,0962	0,2143	0,2083	0,0820	0,1294
	CloudToolkit	0,0569	0,0259	0,0192	0,0429	0,1250	0,0492	0,0532
	LivCloud	0,0797	0,0259	0,0192	0,0143	0,0417	0,0492	0,0383
	SpotifyMod	0,1992	0,3879	0,2885	0,2143	0,2083	0,2459	0,2574
	SUM	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000

PreMigSt		PropMig	SMEfram	CloudMod	CloudToo	LivCloud	SpotifyMod	Weights
	PropMig	0,2830	0,2970	0,2098	0,1731	0,1923	0,3323	0,247917
	SMEfram	0,1415	0,1485	0,3147	0,2885	0,1923	0,1108	0,199373
	CloudModel	0,1415	0,0495	0,1049	0,1731	0,1923	0,1108	0,128676
	CloudToolkit	0,0943	0,0297	0,0350	0,0577	0,1154	0,0665	0,066423
	LivCloud	0,0566	0,0297	0,0210	0,0192	0,0385	0,0475	0,035408
	SpotifyMod	0,2830	0,4455	0,3147	0,2885	0,2692	0,3323	0,322203
	SUM	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000

ValidStake		PropMig	SMEfram	CloudMod	CloudToo	LivCloud	SpotifyMod	Weights
	PropMig	0,3933	0,3440	0,2604	0,2996	0,2188	0,5160	0,338667
	SMEfram	0,0787	0,0688	0,1563	0,0499	0,1563	0,0516	0,093581
	CloudModel	0,0787	0,0229	0,0521	0,0300	0,1563	0,0516	0,065246
	CloudToolkit	0,1966	0,2064	0,2604	0,1498	0,2188	0,0860	0,186333
	LivCloud	0,0562	0,0138	0,0104	0,0214	0,0313	0,0369	0,02831
	SpotifyMod	0,1966	0,3440	0,2604	0,4494	0,2188	0,2580	0,287863
	SUM	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000

LiftEasy		PropMig	SMEfram	CloudMod	CloudToo	LivCloud	SpotifyMod	Weights
	PropMig	0,2857	0,3409	0,2848	0,1667	0,1552	0,3093	0,257092
	SMEfram	0,1429	0,1705	0,2848	0,1667	0,2586	0,1546	0,196341
	CloudModel	0,0952	0,0568	0,0949	0,1667	0,2586	0,1031	0,129229
	CloudToolkit	0,0952	0,0568	0,0316	0,0556	0,0172	0,0619	0,053059
	LivCloud	0,0952	0,0341	0,0190	0,1667	0,0517	0,0619	0,071427
	SpotifyMod	0,2857	0,3409	0,2848	0,2778	0,2586	0,3093	0,292852
	SUM	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000

RewriteCp		PropMig	SMEfram	CloudMod	CloudToo	LivCloud	SpotifyMod	Weights
	PropMig	0,3547	0,3440	0,3908	0,3000	0,2692	0,3409	0,333288
	SMEfram	0,0709	0,0688	0,0434	0,2143	0,1154	0,0682	0,096838
	CloudModel	0,1182	0,2064	0,1303	0,2143	0,2692	0,1136	0,175349
	CloudToolkit	0,0507	0,0138	0,0261	0,0429	0,1154	0,0682	0,052819
	LivCloud	0,0507	0,0229	0,0186	0,0143	0,0385	0,0682	0,035525
	SpotifyMod	0,3547	0,3440	0,3908	0,2143	0,1923	0,3409	0,306181
	SUM	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000

PostMigVal		PropMig	SMEfram	CloudMod	CloudToo	LivCloud	SpotifyMod	Weights
	PropMig	0,3286	0,3030	0,2885	0,2885	0,2917	0,3409	0,306861
	SMEfram	0,1643	0,1515	0,2885	0,2885	0,2083	0,1136	0,202455
	CloudModel	0,0657	0,0303	0,0577	0,0577	0,1250	0,0682	0,067433
	CloudToolkit	0,0657	0,0303	0,0577	0,0577	0,1250	0,0682	0,067433
	LivCloud	0,0469	0,0303	0,0192	0,0192	0,0417	0,0682	0,037594
	SpotifyMod	0,3286	0,4545	0,2885	0,2885	0,2083	0,3409	0,318225
	SUM	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000	1,0000

Teste da consistência

								CriteriaWeights		Weighted Sum Value
DiffServModels	1,00	3,00	3,00	5,00	9,00	3,00	x	0,3668	=	2,538
	0,33	1,00	3,00	5,00	5,00	0,33	x	0,1606	=	1,085
	0,33	0,33	1,00	5,00	3,00	0,20	x	0,1039	=	0,674
	0,20	0,20	0,20	1,00	3,00	0,14	x	0,0502	=	0,304
	0,11	0,20	0,33	0,33	1,00	0,11	x	0,0286	=	0,185
	0,33	3,00	5,00	7,00	9,00	1,00	x	0,2900	=	2,022

								CriteriaWeights		Weighted Sum Value
GenerPlan	1,00	3,00	3,00	5,00	7,00	3,00	x	0,3615	=	2,515
	0,33	1,00	5,00	5,00	7,00	0,33	x	0,2075	=	1,391
	0,33	0,20	1,00	3,00	3,00	0,33	x	0,0961	=	0,603
	0,20	0,20	0,33	1,00	3,00	0,20	x	0,0562	=	0,345
	0,14	0,14	0,33	0,33	1,00	0,14	x	0,0313	=	0,199
	0,33	3,00	3,00	5,00	7,00	1,00	x	0,2475	=	1,778

								CriteriaWeights		Weighted Sum Value
SpecPlan	1,00	3,00	3,00	7,00	5,00	2,00	x	0,3458	=	2,340
	0,33	1,00	3,00	5,00	5,00	0,33	x	0,1759	=	1,223
	0,33	0,33	1,00	5,00	5,00	0,33	x	0,1294	=	0,847
	0,14	0,20	0,20	1,00	3,00	0,20	x	0,0532	=	0,330
	0,20	0,20	0,20	0,33	1,00	0,20	x	0,0383	=	0,238
	0,50	3,00	3,00	5,00	5,00	1,00	x	0,2574	=	1,804

								CriteriaWeights		Weighted Sum Value
PreMigSt	1,00	2,00	2,00	3,00	5,00	1,00	x	0,247917	=	1,603
	0,50	1,00	3,00	5,00	5,00	0,33	x	0,199373	=	1,326
	0,50	0,33	1,00	3,00	5,00	0,33	x	0,128676	=	0,803
	0,33	0,20	0,33	1,00	3,00	0,20	x	0,066423	=	0,402
	0,20	0,20	0,20	0,33	1,00	0,14	x	0,035408	=	0,219
	1,00	3,00	3,00	5,00	7,00	1,00	x	0,322203	=	2,134

								CriteriaWeights		Weighted Sum Value
ValidStake	1,00	5,00	5,00	2,00	7,00	2,00	x	0,338667	=	2,279
	0,20	1,00	3,00	0,33	5,00	0,20	x	0,093581	=	0,618
	0,20	0,33	1,00	0,20	5,00	0,20	x	0,065246	=	0,401
	0,50	3,00	5,00	1,00	7,00	0,33	x	0,186333	=	1,257
	0,14	0,20	0,20	0,14	1,00	0,14	x	0,028310	=	0,176
	0,50	5,00	5,00	3,00	7,00	1,00	x	0,287863	=	2,009

								CriteriaWeights		Weighted Sum Value
LiftEasy	1,00	2,00	3,00	3,00	3,00	1,00	x	0,257092	=	1,704
	0,50	1,00	3,00	3,00	5,00	0,50	x	0,196341	=	1,375
	0,33	0,33	1,00	3,00	5,00	0,33	x	0,129229	=	0,894
	0,33	0,33	0,33	1,00	0,33	0,20	x	0,053059	=	0,330
	0,33	0,20	0,20	3,00	1,00	0,20	x	0,071427	=	0,440
	1,00	2,00	3,00	5,00	5,00	1,00	x	0,292852	=	1,953

								CriteriaWeights		Weighted Sum Value
RewriteCp	1,00	5,00	3,00	7,00	7,00	1,00	x	0,333288	=	2,268
	0,20	1,00	0,33	5,00	3,00	0,20	x	0,096838	=	0,654
	0,33	3,00	1,00	5,00	7,00	0,33	x	0,175349	=	1,192
	0,14	0,20	0,20	1,00	3,00	0,20	x	0,052819	=	0,323
	0,14	0,33	0,14	0,33	1,00	0,20	x	0,035525	=	0,219
	1,00	5,00	3,00	5,00	5,00	1,00	x	0,306181	=	2,091

								CriteriaWeights		Weighted Sum Value
PostMigVal	1,00	2,00	5,00	5,00	7,00	1,00	x	0,306861273	=	1,967
	0,50	1,00	5,00	5,00	5,00	0,33	x	0,202454529	=	1,324
	0,20	0,20	1,00	1,00	3,00	0,20	x	0,067432861	=	0,413
	0,20	0,20	1,00	1,00	3,00	0,20	x	0,067432861	=	0,413
	0,14	0,20	0,33	0,33	1,00	0,20	x	0,037593568	=	0,231
	1,00	3,00	5,00	5,00	5,00	1,00	x	0,318224909	=	2,095

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.59

			CI	CR=CI/RI
	STEP2	STEP3	STEP4	STEP5
Weighted Sum Value				
2,538	6,919095	6,610696	0,122139	0,098499
1,085	6,755454			
0,674	6,491426			
0,304	6,049488			
0,185	6,477578			
2,022	6,971134			

			CI	CR=CI/RI
	STEP2	STEP3	STEP4	STEP5
Weighted Sum Value				
2,515	6,956144	6,603142	0,120628	0,097281
1,391	6,703816			
0,603	6,274095			
0,345	6,148708			
0,199	6,35069			
1,778	7,185401			

			CI	CR=CI/RI
	STEP2	STEP3	STEP4	STEP5
Weighted Sum Value				
2,340	6,767141	6,613351	0,12267	0,098928
1,223	6,952785			
0,847	6,540324			
0,330	6,207225			
0,238	6,204186			
1,804	7,008445			

			CI	CR=CI/RI
	STEP2	STEP3	STEP4	STEP5
Weighted Sum Value				
1,603	6,463964	6,369226	0,073845	0,059553
1,326	6,650422			
0,803	6,238958			
0,402	6,059504			
0,219	6,178622			
2,134	6,623885			

			CI	CR=CI/RI
	STEP2	STEP3	STEP4	STEP5
Weighted Sum Value				
2,279	6,730389	6,570421	0,114084	0,092003
0,618	6,606977			
0,401	6,139286			
1,257	6,744713			
0,176	6,223872			
2,009	6,977286			

			CI	CR=CI/RI
	STEP2	STEP3	STEP4	STEP5
Weighted Sum Value				
1,704	6,627093	6,598853	0,119771	0,096589
1,375	7,0047			
0,894	6,920307			
0,330	6,213059			
0,440	6,159928			
1,953	6,66803			

			CI	CR=CI/RI
	STEP2	STEP3	STEP4	STEP5
Weighted Sum Value				
2,268	6,805285	6,577866	0,115573	0,093204
0,654	6,75201			
1,192	6,796704			
0,323	6,109155			
0,219	6,173359			
2,091	6,830684			

			CI	CR=CI/RI
	STEP2	STEP3	STEP4	STEP5
Weighted Sum Value				
1,967	6,411623	6,320164	0,064033	0,051639
1,324	6,541007			
0,413	6,126903			
0,413	6,126903			
0,231	6,131954			
2,095	6,582597			

Matrizes com prioridades globais

	DiffServM	GenerPlan	SpecPlan	PreMigSt	ValidStak	LiftEasy	RewriteCp	PostMigVal					
PropMig	0,3668	0,3615	0,3458	0,2479	0,3387	0,2571	0,3333	0,3069	x	0,023945	=	0,318	The Model that will be developed
SMEfram	0,1606	0,2075	0,1759	0,1994	0,0936	0,1963	0,0968	0,2025	x	0,30093	=	0,182	
CloudModel	0,1039	0,0961	0,1294	0,1287	0,0652	0,1292	0,1753	0,0674	x	0,089785	=	0,104	
CloudToolkit	0,0502	0,0562	0,0532	0,0664	0,1863	0,0531	0,0528	0,0674	x	0,201449	=	0,075	
LivCloud	0,0286	0,0313	0,0383	0,0354	0,0283	0,0714	0,0355	0,0376	x	0,118776	=	0,036	
SpotifyMod	0,2900	0,2475	0,2574	0,3222	0,2879	0,2929	0,3062	0,3182	x	0,068383	=	0,286	The Best Model to use as inspiration
										0,045869	=		
										0,150863	=		

Appendix C – First Stakeholders Meeting Record

Meeting/Project Name	Service Cloud Migration Planning with Stakeholders		
Date of Meeting	29/03/2022	**Time**	11AM
Meeting Facilitator	Pedro Quintã	**Location**	Blip Oporto Office

Meeting Objectives
Discuss SBS migration from Private Cloud to Public Cloud

Attendees	
Name	**Position**
Pedro Quintã	Backend Developer
Miguel Veiga	Principal Backend Developer
	Delivery Manager
	Backend Developer

Meeting Agenda	
Topics / Discussion Notes	**Discussed led by**
Database Questions: The discussion was centred on the database and whether or not the data in the database needed to be migrated as well. It was concluded that it would not be necessary to copy the data and it would simply be necessary to create a database in the new cloud	Pedro Quintã
Service Configurations: Issues like current system configurations (CPU, memory, disk, number of machines, between others) were discussed and configuration options for the new deployment were also discussed. Up to now, Chef Cookbooks were used to configure different environments and it will be necessary to find an option to perform the same kind of configurations per environment	Pedro Quintã
Performance and Metrics: Topics such as performance and the possibility of having a base metrics dashboard for future development by the team responsible for the SBS were highlighted	Backend Developer responsible for SBS

Action Items	
Actions	**Responsible**
Create a clear Database on AWS	Pedro Quintã

Configure CDKs for Service, Apache Kafka and Database	Pedro Quintã
Add Grafana Cloud configurations while deploying service CDK	Pedro Quintã

Appendix D – Second Stakeholders Meeting Record

Meeting/Project Name	Discussion meeting about SBS Service		
Date of Meeting	10/05/2022	**Time**	10AM
Meeting Facilitator	Pedro Quintã	**Location**	Blip Oporto Office

Meeting Objectives
Discuss SBS questions and logging/metrics implementation

Attendees	
Name	**Position**
Pedro Quintã	Backend Developer
Miguel Veiga	Principal Backend Developer
	Delivery Manager
	Backend Developer

Meeting Agenda	
Topics / Discussion Notes	**Discussed led by**
Service Questions: The discussion was centred on the Akka System and how this type of service work. The functioning of the service in the current infrastructure was discussed and some specificities were noted to be applied in the implementation.	Pedro Quintã
Logging System: In the current implementation on AWS, the services were using CloudWatch Logging and the need to go for a more complete choice like Grafana Cloud was questioned.	Pedro Quintã
Metrics Implementation and System: Given that the approach was to use Grafana Cloud for logging, the choice was straightforward for the service that would collect metrics from the Service. It was decided that the development team that currently maintains the service, would further develop the metrics and apply the current ones to the service on AWS	Pedro Quintã

Action Items	
Actions	**Responsible**
Apply specific Akka Configurations on CDK	Pedro Quintã
Create Grafana Log Driver implementation on CDK	Pedro Quintã

Appendix E – Stakeholders Survey

Question 1 - Does the implemented solution enable a better understanding of the migration process to the public cloud?

Question 2 – Does the implemented solution facilitate the migration process?

Question 3 - Does the methodology cover the entire migration process?

Question 4 - Is the maintenance process well defined?

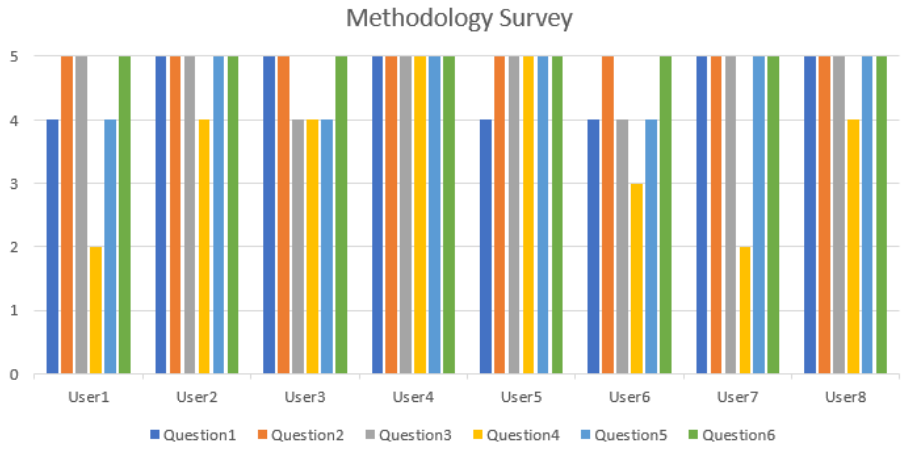
Question 5 - Do the base infrastructure models (CDK) for this methodology help in application development?

Question 6 - Taking into account the target public of the developed model, is the methodology fundamental in the migration of services?

Possible Options: Totally Disagree (1), Partially Disagree (2), Indifferent (3), Partially Agree (4) and Totally Agree (5)

Results:

	User1	User2	User3	User4	User5	User6	User7	User8
Question1	4	5	5	5	4	4	5	5
Question2	5	5	5	5	5	5	5	5
Question3	5	5	4	5	5	4	5	5
Question4	2	4	4	5	5	3	2	4
Question5	4	5	4	5	5	4	5	5
Question6	5	5	5	5	5	5	5	5



Appendix F – Shapiro-Wilk Normality Test

Question 1:

Sorted Sample	Order	(xi -xmedio)^2	Lower Half	Top Half	i	N-i +1	a(N-i+a)	Difference	a(N-i+a) * Difference	b	Statistics	alfa	Tabulated Value
4	1	0,390625	4	5	1	8	0,6062	1	0,6062	1,0969	0,64170113	0,05	0,818
4	2	0,390625	4	5	2	7	0,3164	1	0,3164	1,0969			
4	3	0,390625	4	5	3	6	0,1743	1	0,1743	1,0969			
5	4	0,140625	5	5	4	5	0,0561	0	0	1,0969			
5	5	0,140625											
5	6	0,140625											
5	7	0,140625											
5	8	0,140625											

Question 2:

Sorted Sample	Order	(xi -xmedio)^2	Lower Half	Top Half	i	N-i +1	a(N-i+a)	Difference	a(N-i+a) * Difference	b	Statistics	alfa	Tabulated Value
5	1	0	5	5	1	8	0,6062	0	0	0	#DIV/0!	0,05	0,818
5	2	0	5	5	2	7	0,3164	0	0	0			
5	3	0	5	5	3	6	0,1743	0	0	0			
5	4	0	5	5	4	5	0,0561	0	0	0			
5	5	0											
5	6	0											
5	7	0											
5	8	0											

Question 3:

Sorted Sample	Order	(xi -xmedio)^2	Lower Half	Top Half	i	N-i +1	a(N-i+a)	Difference	a(N-i+a) * Difference	b	Statistics	alfa	Tabulated Value
4	1	0,5625	4	5	1	8	0,6062	1	0,6062	0,9226	0,5674605	0,05	0,818
4	2	0,5625	4	5	2	7	0,3164	1	0,3164	0,9226			
5	3	0,0625	5	5	3	6	0,1743	0	0	0,9226			
5	4	0,0625	5	5	4	5	0,0561	0	0	0,9226			
5	5	0,0625											
5	6	0,0625											
5	7	0,0625											
5	8	0,0625											

Question 4:

Sorted Sample	Order	(xi -xmedio)^2	Lower Half	Top Half	i	N-i +1	a(N-i+a)	Difference	a(N-i+a) * Difference	b	Statistics	alfa	Tabulated Value
2	1	2,640625	2	5	1	8	0,6062	3	1,8186	2,9421	0,87655214	0,05	0,818
2	2	2,640625	2	5	2	7	0,3164	3	0,9492	2,9421			
3	3	0,390625	3	4	3	6	0,1743	1	0,1743	2,9421			
4	4	0,140625	4	4	4	5	0,0561	0	0	2,9421			
4	5	0,140625											
4	6	0,140625											
5	7	1,890625											
5	8	1,890625											

Question 5:

Sorted Sample	Order	(xi -xmedio)^2	Lower Half	Top Half	i	N-i +1	a(N-i+a)	Difference	a(N-i+a) * Difference	b	Statistics	alfa	Tabulated Value
4	1	0,390625	4	5	1	8	0,6062	1	0,6062	1,0969	0,64170113	0,05	0,818
4	2	0,390625	4	5	2	7	0,3164	1	0,3164	1,0969			
4	3	0,390625	4	5	3	6	0,1743	1	0,1743	1,0969			
5	4	0,140625	5	5	4	5	0,0561	0	0	1,0969			
5	5	0,140625											
5	6	0,140625											
5	7	0,140625											
5	8	0,140625											

Question 6:

Sorted Sample	Order	(xi -xmedio)^2	Lower Half	Top Half	i	N-i +1	a(N-i+a)	Difference	a(N-i+a) * Difference	b	Statistics	alfa	Tabulated Value
5	1	0	5	5	1	8	0,6062	0	0	0	#DIV/0!	0,05	0,818
5	2	0	5	5	2	7	0,3164	0	0	0			
5	3	0	5	5	3	6	0,1743	0	0	0			
5	4	0	5	5	4	5	0,0561	0	0	0			
5	5	0											
5	6	0											
5	7	0											
5	8	0											

Appendix G – Service Changes (Step by Step)

To start this process, we will need to have some prerequisites and some changes in the service code:

1. The service must be containerized. To do so, Dockerfile needs to be created and it will use the project jar to create a simple image for the project and it is also possible to add some JVM options. The following example will use akka-application as an example service name:

```
FROM eclipse-temurin:11.0.16_8-jre
RUN apt-get update && apt-get install -y curl

ADD target/akka-application-*.jar /opt/service/fbo-application.jar
ADD src/main/resources/application.conf /opt/service/
WORKDIR /opt/service/
ENTRYPOINT java -jar -Dconfig.file=application--$ENVIRONMENT.conf akka-application.jar
```

2. A variable $ENVIRONMENT is used in the above code, this variable will be important later on.
3. Add the following maven dependencies to the application POM:
   a. **com.typesafe.akka:akka-actor_${scala.binary.version}** -> version is the same as the akka version
   b. **com.typesafe.akka:akka-cluster-sharding_${scala.binary.version**} -> version is the same as the akka version
   c. **com.amazonaws:aws-java-sdk-core** -> aws-java-sdk-core version
   d. **com.lightbend.akka.management:akka-management-cluster-bootstrap_${scala.binary.version}** -> akka management version
   e. **com.typesafe.akka:akka-slf4j_${scala.binary.version}** -> version is the same as the akka version
   f. **com.typesafe.akka:akka-discovery_${scala.binary.version}** -> version is the same as the akka version
   g. **com.lightbend.akka.discovery:akka-discovery-aws-api_${scala.binary.version}**-> akka management version
   h. **com.lightbend.akka.management:akka-management_${scala.binary.version}** -> akka management version
   i. **com.lightbend.akka.management:akka-management-cluster-http_${scala.binary.version}** -> akka management version
   j. **com.typesafe.akka:akka-cluster_${scala.binary.version}** -> version is the same as the akka version
   k. **com.typesafe.akka:akka-stream_${scala.binary.version}** -> version is the same as the akka version
4. Sometimes there may be errors from different versions coming within the above dependencies. To check that, the command “mvn dependency:tree -Dincludes=” can help by listing the dependency tree for the problematic dependency. If needed, add <exclusion> to the dependency that brings the problematic version with it.

5. Make sure the Actor System Configuration is starting the AkkaManagement and the ClusterBootstrap. Something similar to this:

```
// Akka Management hosts the HTTP routes used by bootstrap
management = AkkaManagement.get(system);
management.start();

// Starting the bootstrap process needs to be done explicitly
ClusterBootstrap.get(system).start();
```

6. Create an application.conf file for each service environment (application-qa.conf, application-prd.conf,…). As an example, below there is an application-local.conf for local testing and a base application.conf for AWS:

```
application-akka {
  akka.http.host-connection-pool.response-entity-subscription-timeout = 3 seconds

  clustering {
      ip = "application-app"
      ip = ${?CLUSTER_IP}

      port = 25520
      port = ${?CLUSTER_PORT}

      seed-ip = "127.0.0.1"
      seed-ip = ${?CLUSTER_IP}
      seed-ip = ${?SEED_PORT_1600_TCP_ADDR}

      seed-port = 25520
      seed-port = ${?SEED_PORT_1600_TCP_PORT}

      cluster.name = "application-as"
    }

  akka {
    loglevel = DEBUG

    actor {
      provider = cluster
    }

    remote.artery {
       enabled = on
       canonical {
         hostname = ${application-akka.clustering.ip}
         port = ${application-akka.clustering.port}
       }
    }

    cluster {
        seed-nodes = ["akka://"${application-akka.clustering.cluster.name}"@"${application-
akka.clustering.seed-ip}":"${application-akka.clustering.seed-port}]
        min-nr-of-member = 2
        sharding {
            number-of-shards = 30
        }
        split-brain-resolver {
            stable-after = 1s
            down-all-when-unstable = off
        }
    }

    management {
        http {
          hostname = 127.0.0.1
          port = 8558
          bind-hostname = 0.0.0.0
          bind-port = 8558
        }
    }
  }
}
```

7. Base application.conf for AWS:

```
application-akka {
  akka {
    loglevel = INFO

    actor {
      provider = akka.cluster.ClusterActorRefProvider
    }

    remote.artery {
       enabled = on
       canonical {
         port = 2551
       }
    }

    cluster {

        # Disable seed-nodes to use cluster bootstrap for ECS discovery in a dynamic environment
        seed-nodes = []

        min-nr-of-member = 2
        sharding {
            number-of-shards = 30
        }

        # Used to shutdown a node if it can't join the cluster in the given period
        # ECS will than use the alive probe to restart the faulty node
        # shutdown-after-unsuccessful-join-seed-nodes = 30s

        # Used to resolve network partitions and down unreachable nodes
        # downing-provider-class = "akka.cluster.sbr.SplitBrainResolverProvider"

        # Split Brain Resolver strategies
        #
        # Uncomment to override default strategy settings : keep-majority (default)
        # Select one of the available strategies (see descriptions below):
        #   static-quorum, keep-majority, keep-oldest, down-all, lease-majority
        # see also https://doc.akka.io/docs/akka/current/split-brain-resolver.html#strategies
        # split-brain-resolver {
        #   active-strategy = keep-majority
        # }
    }

    management {
        cluster.bootstrap {

          ## On initial deployment use the default new-cluster-enabled = on
          ## - Also if cluster configuration is changed then new-cluster-enabled should be set to 'on'
          ## Following the initial deployment it is recommended to set new-cluster-enabled = off
          new-cluster-enabled = on

          contact-point {
               fallback-port = 8080
               # If some discovered seed node will keep failing to connect for specified period of time,
               # it will initiate rediscovery again instead of keep trying.
               probing-failure-timeout = 15 seconds
          }

          contact-point-discovery {
            ## AWS ECS Service Name
            ## used by discovery service to query ECS metadata service and obtain the IP of the other nodes
            ## service-name = ${ECS_SERVICE_NAME}
            service-name = ${ECS_SERVICE_NAME}

            ## Setting to the exact number of nodes the initial startup of the cluster will be done
            required-contact-point-nr = 2

            # Timeout for getting a reply from the service-discovery subsystem
            resolve-timeout = 600 seconds
          }
        }
        http {
          port = 8080
          bind-hostname = 0.0.0.0
          bind-port = 8080
        }
    }

    ## Enable ClusterBootstrap
    extensions = ["akka.management.cluster.bootstrap.ClusterBootstrap"]

    discovery {
       ## Enable AWS ECS Service discovery for awsvpc networks
       method = aws-api-ecs
       aws-api-ecs {
         ## AWS ECS Cluster Name
         ## cluster = ${ECS_CLUSTER_NAME}
         cluster = ${ECS_CLUSTER_NAME}
       }
    }
  }
}
```

8. To test locally, you can use a docker container that starts all the cluster and dependencies:

```
version: '3.8'

services:
  application-zookeeper:
    image: 'wurstmeister/zookeeper:latest'
    hostname: application-zookeeper
    ports:
      - "2181:2181"
  application-kafka:
    image: 'wurstmeister/kafka:latest'
    ports:
      - "9092:9092"
      - "9094:9094"
    environment:
      KAFKA_ZOOKEEPER_CONNECT: application-zookeeper:2181
      KAFKA_CREATE_TOPICS: recommendation.pb.stream.pp:3:1,sb_market_stream_ng_topic:3:1
      KAFKA_LISTENERS: INSIDE://0.0.0.0:9092,OUTSIDE://0.0.0.0:9094
      KAFKA_ADVERTISED_LISTENERS: INSIDE://application-kafka:9092,OUTSIDE://localhost:9094
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: INSIDE
    depends_on:
        - application-zookeeper

  application-mysql:
    image: mysql:latest
    hostname: application-mysql
    restart: always
    environment:
      MYSQL_DATABASE: 'applicationdb'
      MYSQL_USER: 'application'              # So you don't have to use root, but you can if you like
      MYSQL_PASSWORD: 'pass'          # You can use whatever password you like
      MYSQL_ROOT_PASSWORD: 'toor'     # Password for root access
    ports:
      - '3306:3306'                   # <Port exposed> : < MySQL Port running inside container>
    volumes:
      - applicationdb:/var/lib/mysql         # Where our data will be persisted
    container_name: 'application-mysql'

  application-app-node-1:
    container_name: application-app-node-1
    hostname: application-app-node-1
    build: application-application/.
    image: sportsbook/application-application
    ports:
      - "25521:25520"
      - "8551:8558"
    environment:
      ENVIRONMENT: local
      CLUSTER_PORT: 25520
      CLUSTER_IP: application-app-node-1
      SEED_PORT_1600_TCP_ADDR: application-app-node-1
    restart: on-failure
    shm_size: "2gb"
    depends_on:
      - application-zookeeper
      - application-kafka
      - application-mysql

  application-app-node-2:
    container_name: application-app-node-2
    hostname: application-app-node-2
    build: application-application/.
    image: sportsbook/application-application
    ports:
      - "25522:25520"
      - "8552:8558"
    environment:
      ENVIRONMENT: local
      CLUSTER_PORT: 25520
      CLUSTER_IP: application-app-node-2
      SEED_PORT_1600_TCP_ADDR: application-app-node-1
    restart: on-failure
    shm_size: "2gb"
    depends_on:
      - application-zookeeper
      - application-kafka
      - application-mysql

  application-app-node-3:
    container_name: application-app-node-3
    hostname: application-app-node-3
    build: application-application/.
    image: sportsbook/application-application
    ports:
      - "25523:25520"
      - "8553:8558"
    environment:
      ENVIRONMENT: local
      CLUSTER_PORT: 25520
      CLUSTER_IP: application-app-node-3
      SEED_PORT_1600_TCP_ADDR: application-app-node-1
    restart: on-failure
    shm_size: "2gb"
    depends_on:
      - application-zookeeper
      - application-kafka
      - application-mysql

volumes:
  applicationdb:                          # Names our volume
```

9. Compile, Create Docker Image and Send image to ECR:
    a. **mvn clean install -Dmaven.test.skip=true**
    b. **docker build -t sportsbook/akka-application .** -> to build the Dockerfile
    c. **docker tag akka-application:latest $ECR_CONNECTION_URL:$VERSION**
    d. **docker push $ECR_CONNECTION_URL:$VERSION**

Akka CDK Changes and Points to be taken into account

For Akka ECS deployment, you must create a general ECS cluster cdk code and add some specific configurations for ECS Service Discovery.

1. Add ingress and egress rules for the service port. Example: 8080

```
sg.addIngressRule(Platform.prefixList('ppb-network'), Port.tcp(8080)); // Service Port
sg.addEgressRule(Platform.prefixList('ppb-network'), Port.tcp(8080)); // Service Port
```

2. Open TCP and UDP (in case you are using “remote.artery.enabled=true” in the application.conf) ports internally between containers:

```
sg.connections.allowInternally(Port.tcp(8080), 'Allows traffic for Akka Management probes')
sg.connections.allowInternally(Port.udp(2551), 'Akka UDP');
```

3. When defining the task definition for the container, take into account the CPU and memory required, you must have the CPU and memory needed to start the service.
4. Specify a container healthcheck container endpoint:

```
healthCheck: {
  command: [ "CMD-SHELL", "curl --request GET 'http://0.0.0.0:8080/cluster/members/' || exit 1" ],
  startPeriod: cdk.Duration.seconds(60),
  timeout: cdk.Duration.seconds(30),
},
```

5. Be sure to set the container port to the HTTP port. Example:8080
6. You must inject an environment variable called ENVIRONMENT that will set the application.conf file that will be used to start the service
7. There are two variables (ECS_SERVICE_NAME and ECS_CLUSTER_NAME) that must be generated in the CDK code and injected as environment variables