

AVISI

Graduation Report

Kubernetes multi-cluster & multi-cloud architectures

How can Avisi provide the customers of its managed platform (PaaS) with full control over the geographical location and cloud providers hosting their applications and data?

Author: Ivan Shishkalov

Study: Saxion University of Applied Sciences, HBO-ICT > ITSM

Student number: 450037

Graduation teacher: Esther Hageraats

Company: Avisi Cloud B.V.

Company supervisor: Thomas Kooi

Version: 1.0

Date: 03-04-2022

Acknowledgements

I have received a great deal of guidance and support throughout this graduation project.

First, I would like to thank Jeroen Veldhorst and Thomas Kooi for the incredible opportunity to work on such a novel problem for my graduation. I was able to learn a lot about cutting-edge technologies in IT infrastructure.

I would like to give my special regards to my company supervisor, Thomas Kooi, and my colleagues from AME for welcoming me to the team and providing the technical assistance and practical suggestions I needed to complete this project.

Further, I would like to express my special thanks to my graduation teacher, Esther Hageraats. Her expertise and continuous feedback were invaluable in shaping this research, establishing the right methodology and bringing the quality of the report to a higher level.

In addition, I would like to thank the people behind the Ligo project. This research is greatly inspired by their work which helped me the most in understanding the subject of Kubernetes multi-cluster from different perspectives.

Furthermore, I would like to express my deepest gratitude to my parents for their everlasting support and selfless efforts. Without them, I would not be receiving this education and delivering this paper.

Finally, I am very grateful to my friend, Tien Thai, for encouraging me all the way and providing a healthy degree of distractions that kept me sane during this period.

Abstract

Kubernetes adoption has grown drastically since its release in 2014. Today, Kubernetes has become the de-facto standard to deploy and orchestrate containerized applications. Avisi Managed Environments (AME) is a cloud-agnostic platform that provides fully managed Kubernetes clusters that host mission-critical applications and data.

There are plans to grow the platform and attract more enterprise customers. However, AME presents limitations for certain use cases such as jurisdiction compliance, disaster recovery, high availability, multi-/hybrid-cloud strategies and more. To solve the above use cases, the Kubernetes environments need to stretch beyond the boundaries of a single region or cloud provider. It is not practical to stretch a single Kubernetes cluster, however, multi-cluster architectures have the potential to address the issue.

Kubernetes multi-cluster architectures introduce primarily two types of challenges: network connectivity and orchestration of workloads across clusters. While networking in a single cluster has been standardized, the solutions that extend it to multi-clusters are novel and unexplored. This graduation project aims to evaluate a wide range of multi-cluster connectivity solutions and propose an architecture that meets the requirements and accounts for the constraints of the current platform.

A literature study focused primarily on the Cloud Native Computing Foundation (CNCF) resources helped to determine nine potential solutions. The research shows that most projects have not yet reached a production-ready status and that there is no single solution to satisfy all AME use cases. Multi-criteria analysis and prototyping led to three final architecture proposals based on Linkerd, Ligo, and NSM.

Linkerd is the simplest architecture that enables direct L7 connectivity between services in different clusters which covers most generic use cases. Ligo can flatten L3 networking across AME clusters using secure VPN tunnels. Moreover, Ligo architecture provides multi-cluster orchestration capabilities that drastically simplify deploying to and operating a multi-cluster environment. NSM is best-suited for scenarios when applications require lower-level network features or non-standard protocols that use Ethernet/IP payloads at L2/L3

The proposed designs were tested with proof-of-concept using test Kubernetes clusters and mock applications. Further investigation is required to evaluate proposed architectures with real applications.

Table of Contents

1	Introduction.....	6
2	Problem Analysis.....	7
2.1	Problem Description	7
2.2	Justification	7
2.3	Problem Statement	8
2.4	Research Questions and Methodology Overview	9
2.5	Goal	11
3	Avisi Managed Environments.....	12
3.1	Methodology.....	12
3.2	AME – the foundation of Avisi Cloud	13
3.3	Business Context	14
3.4	Architecture Overview	15
3.5	Conclusion and Design Constraints.....	18
4	The Limits of the AME platform	19
4.1	Methodology.....	19
4.2	Problem – Deep Dive.....	19
4.3	Problem – Core.....	20
5	Stakeholders and User Stories	22
5.1	Methodology.....	22
5.2	Stakeholders – Shortlist	22
5.3	Summary of Stakeholder Interviews	23
5.4	User Stories	23
6	Technical Research: K8s Multi-cluster	27
6.1	Methodology.....	27
6.2	Provider-agnostic and Multi-cloud K8s	28
6.3	K8s Networking and Multi-cluster Connectivity	29
6.4	Design Principles	35
7	Conceptual Design.....	37
7.1	Kubernetes Concepts	37

7.2	Conceptual Multi-cluster Architectures.....	40
7.3	Multi-cluster Orchestration.....	43
7.4	Conclusion.....	44
8	Technical Design.....	45
8.1	Introduction.....	45
8.2	Proposed Architecture 1 – L7 Multi-cluster connectivity with Linkerd.....	45
8.3	Proposed Architecture 2 – L3 Multi-cluster connectivity with Ligo.....	46
8.4	Proposed Architecture 3 – L2/L3 Point-to-Point Multi-cluster connectivity with NSM.....	47
8.5	Conclusion.....	49
9	Proof of Concept.....	50
9.1	Prototyping Environment.....	50
9.2	Linkerd.....	50
9.3	Ligo.....	50
9.4	NSM.....	51
10	Advice on Multi-cluster Challenges.....	52
10.1	Complexity.....	52
10.2	Security.....	54
10.3	Observability.....	55
11	Conclusions and Reflection.....	56
11.1	Product Reflection.....	57
11.2	Self-Reflection.....	58
	List of Figures and Tables.....	59
	List of Abbreviations.....	60
	Reference List.....	61
	Versions.....	65

1 Introduction

Avisi Managed Environments (AME) is a foundational product of Avisi Cloud. It is a cloud-agnostic platform that provides fully managed Kubernetes clusters. The AME platform solves the most pressing problems faced by Kubernetes users: core infrastructure, cluster upgrades, and day 2 operations (e.g., monitoring, optimizing, performing backups). This graduation assignment was created from a desire to enrich the capabilities of the current platform, namely, through multi-cluster architectures.

Kubernetes community and existing research indicate that the number of Kubernetes clusters used by organizations is rapidly growing and multi-cluster environments are becoming more and more common. This trend is also confirmed by the emergence of new open-source projects such as Submariner, NSM, and Liqo that aim to enable direct networking and simplify deployments across multiple clusters. At the same time, neither networking nor orchestration in multi-cluster environments has been standardized by Kubernetes as of today. While solutions are diverse, they serve different use cases and use different underlying technology. Therefore, this research was essential as it aims specifically at the multi-cluster use cases relevant to Avisi. Moreover, it evaluates the widest range of possible solutions and focuses on seamless integration with the company's current cloud-agnostic Kubernetes platform – Avisi Managed Environments. Avisi recognizes specific use cases and potential benefits from enabling multi-cluster architectures, however, it is not clear how to achieve them. As agreed with the company, the primary scope of this research was on networking, however, the orchestration was also addressed.

An iterative approach (RUP) was used which resulted in overlaps between research questions. The structure of this document follows the order of questions defined in chapter 2.4. Chapter 2 introduces the overarching problem and opportunity studied in this research, along with a methodology overview and goal. Then, Chapter 3 focuses on understanding the current Avisi Managed Environments platform, its architecture and design constraints to be considered. Followed by Chapter 4 that delves deeper into the technical aspects of the problem; it talks about the primary (networking) and secondary (orchestration) challenges that go into Kubernetes multi-cluster architectures. Chapter 5 talks about the requirements identified through Stakeholder Analysis. All technical research including a multi-criteria analysis of various Kubernetes multi-cluster connectivity solutions is condensed in Chapter 6. In addition, the chapter defines essential design principles. The findings are then used to present a conceptual design in Chapter 7 and Technical Design in Chapter 8. Subsequently, Chapter 9 briefly discusses the prototypes created as PoCs to validate the proposed designs. Finally, Chapter 10 addresses the challenges of multi-cluster architectures and provides advice on how to handle them in the future.

2 Problem Analysis

This chapter introduces the overarching business problem and opportunity discovered through the first conversations with Avisi. Based on the defined problem, the main research question has been formulated in 2.4.1 along with the sub-questions 2.4.2 to navigate the research in a structured way. The analysis of the problem continues in chapter 4, delving into the technical challenges of Kubernetes (K8s) on the path to achieving a multi-cloud architecture.

2.1 Problem Description

Avisi Cloud offers a fully managed provider-agnostic Kubernetes platform called Avisi Managed Environments (AME). The platform is designed to run containerized workloads safely and reliably; AME takes charge of ensuring all environments are available, fast, and secure allowing customers to have more environments and fewer Site-Reliability-Engineers. Furthermore, it equips organizations with a set of fine-tuned out-of-the-box tools such as observability, logging, and alerting, allowing them to have insights into the performance of their applications and focus on improvements without any operational toil. More details about the platform can be found in chapter 3.2.

Today, AME wants to grow and attract new customers – enterprises with large-scale mission-critical applications. Organizations like this have special demands for jurisdiction compliance, high availability, resiliency, and disaster recovery. To effectively deal with these demands, companies should be able to deploy, scale, and migrate their applications and data freely across geographies and providers. In addition, many enterprises adopt a multi-cloud strategy – it allows to get the best out of multiple vendors and avoid vendor lock-in. Currently, AME cannot offer this degree of choice because only two geographical regions and few infrastructure vendors are supported. Moreover, current K8s clusters are siloed within one region (no multi-region capability) and one provider (no multi-cloud capability). To fulfil the use cases above, workload distribution needs to go beyond a single region and provider.

Recently, the industry has taken the next step in the adoption of Kubernetes – multi-cluster architectures. The CTO and the Platform Architect of Avisi consider Kubernetes multi-cluster architectures an opportunity to address the enterprise use cases and make it a unique selling point for AME. However, the company does not have sufficient knowledge about such an architecture, the tools that enable it, how to integrate it into the current AME architecture and what challenges it might bring.

2.2 Justification

According to the Flexera 2021 State of the Cloud report, 92 percent of enterprises reported having a multi-cloud strategy [1]. Organizations opt for this strategy in pursuit of various goals, be it optimization of cloud spend, achieving the lowest possible latency, global distribution, being able to quickly react to changes in laws and regulations, using provider-specific features of different public

clouds, and more. Use cases evaluation (Appendix A, chapter 1.5) showed that there are at least four critical use cases that AME cannot effectively address in its current state. All these use cases could potentially be solved with K8s multi-cluster.

Use Case 1. Jurisdiction Compliance. Data residency and processing laws can require compute and storage to live within a specific region. For large organizations that operate internationally, it might be required for infrastructure to be deployed in multiple geographic regions or cloud providers. Currently, AME wouldn't be able to achieve this, but multi-cluster could make it possible.

Use Case 2. Disaster Recovery. Customer environments are comprised of single clusters running workloads. K8s single clusters are distributed but only inside a single region (multiple availability zone). If there's an infrastructure failure within an entire region, the cluster "goes down" and all services become unavailable. With current architecture, AME wouldn't be able to quickly run somewhere else. With multi-cluster, however, traffic could failover to another cluster.

Use Case 3. Multi-region Availability and Low Latency (Global Distribution). Large-scale enterprise applications that operate globally need exceptional measures to increase availability and lower latency to provide the best user experience. For instance, one of the customers of Avisi wants to expand their business to Africa. To achieve this, the distribution of workloads must go beyond a single region and become global.

Use Case 4. Provider-specific features. Many enterprises adopt a multi-cloud strategy to take advantage of the best features from multiple providers, e.g., advanced AI or GPU capabilities. Since AME is a K8s-based platform, multi-cloud cannot be achieved without a multi-cluster topology.

To sum up, K8s multi-cluster architecture has the potential to create a unique selling point for AME and bring more customers. On the contrary, not carrying out this project may result in negative consequences for the business: the topic is actively gaining traction in the industry and missing out on this topic for Avisi means falling behind the competition and becoming less attractive to new customers. Furthermore, the current customers of AME might face vendor lock-in which can result in higher costs, loss of control over their data, and inability to easily transfer their workloads to another provider.

2.3 Problem Statement

Based on the previous information, an overarching problem can be outlined. The current state of the AME platform cannot provide target customers with sufficient flexibility and control over the geographical location and cloud providers, and therefore cannot effectively address the following critical use cases: jurisdiction compliance, disaster recovery, global distribution, and provider-specific features (of multiple vendors). A novel type of K8s architecture – multi-cluster – is a big opportunity for AME to address the use cases above but it is yet unexplored. A comprehensive investigation is required into K8s multi-cluster architectures and tools.

2.4 Research Questions and Methodology Overview

Based on the overarching problem, the research will focus on investigating the concepts of K8s multi-cloud and multi-cluster architectures to allow Avisi to make the AME platform more flexible. To navigate the research, the following main and sub-questions have been formulated.

2.4.1 Main Research Question

The main research question is “How can Avisi provide the customers of its managed platform (PaaS) with full control over the geographical location and cloud providers hosting their applications and data?”

2.4.2 Sub-Questions

The table below lists the sub-questions, methods used and references to the chapters where the sub-questions are elaborated.

#	Sub-question	Methods	Chapter Ref.
1	What is the Avisi Managed Environments (AME) platform? <i>Objective: Current situation, Design Constraints</i>	<ul style="list-style-type: none">• Document Analysis• Observation• Interviews• Literature Study	3
2	Why does the current architecture limit AME in supporting more cloud providers and geographic locations? <i>Objective: Technical Problem Analysis, Root Cause(s)</i>	<ul style="list-style-type: none">• Problem Analysis• Document Analysis• Interviews• Prototyping	4
3	What requirements do the stakeholders at the company have for the future solution? <i>Objective: User Stories and Requirements</i>	<ul style="list-style-type: none">• Stakeholder Analysis• Requirement Analysis• Interviews• Brainstorming• Literature Study	5.4
4	What current trends, best practices, and reference architectures exist around building multi-cluster Kubernetes environments for Avisi use cases? <i>Objective: Design Principles and Reference Architectures</i>	<ul style="list-style-type: none">• Literature Study• Community Research• Expert Interview• Prototyping• Multi-criteria Analysis	6.4
5	How can the discovered best practices and reference architectures be applied to the AME platform conceptually? <i>Objective: Conceptual Design</i>	<ul style="list-style-type: none">• Literature Study• Brainstorming• IT Architecture Sketching	7
6	How to achieve the desired AME architecture with respect to all system requirements and the conceptual design? <i>Objective: Technical Design</i>	<ul style="list-style-type: none">• IT Architecture Sketching• Guideline Conformity Analysis	8

		<ul style="list-style-type: none"> • Peer Review 	
7	<p>How can the implementation of the Technical Design be automated in the future?</p> <p><i>Objective: Improved Technical Design</i></p>	<ul style="list-style-type: none"> • Literature Study • Community Research • Prototyping • Peer Review 	Appx. B
8	<p>How to prove that the resulting Technical Design is a working solution to the problem?</p> <p><i>Objective: Proof-of-Concept (final prototype)</i></p>	<ul style="list-style-type: none"> • Prototyping • Testing (Usability, System) • Peer Review 	9 & Appx. E
9	<p>What are the challenges associated with complexity, security, and observability in a Kubernetes multi-cluster environment, and what could be done to overcome these challenges?</p> <p><i>Objective: Advisory Report</i></p>	<ul style="list-style-type: none"> • Literature Study • Community Research • Observation • Peer Review 	10

Table 1. Sub-questions, methods, and chapter references

2.4.3 Methodology Overview

The researched concepts are novel; therefore, this graduation project is very experimental in its nature and requires an appropriate and flexible process. The research will be divided into 4 iterative phases of the Rational Unified Process (RUP) with the focus on continuous improvement through the DOT methods listed above.

Inception phase will use iterations of document analysis, observation, interviews with stakeholders and prototyping to better understand the current architecture and problems within it. (Research sub-questions 1 and 2).

Elaboration phase will focus on understanding the needs of Avisi. Iterations of stakeholder interviews and brainstorms with the team will keep user stories emergent and evolving. Furthermore, during this phase the first steps will be made in technical literature study and community research, leading to a set of design principles and a Conceptual Design. (Research sub-questions 3, 4 and 5).

Construction is the phase of continuous trial and error approach. During this phase, most of the work on the Technical Design and prototyping will happen with the help of methods such as literature study, IT architecture sketching, multi-criteria analysis, peer review and testing. (Research sub-questions 6, 7 and 8).

Transition is the final phase of the project; it will focus on evaluating the Technical Design and addressing the challenges that the company might face implementing the design in production and ways to overcome them. To assist this phase, methods like peer reviews and community research will be used. (Research sub-question 9).

One of the key methods in this project is prototyping. Small prototypes will be created throughout the project to understand how the current AME architecture works, test various multi-cluster interconnection techniques, discover technical limitations or possibilities that are not documented

and cannot be found in literature. The purpose, procedures and lessons learned will be documented for each prototype allowing to reuse knowledge and create better prototypes each iteration. Eventually, the small prototypes will lead to the creation of the final Proof of Concept prototype that will be demonstrated at the end of the project and prove that the resulting Technical Design works in practice.

2.5 Goal

This project aims to investigate the different types of K8s multi-cluster architecture and connectivity solutions and design an architecture that fulfils the use cases relevant to Avisi. Further, another important goal is to address the challenges that come with using such an architecture and how to overcome them.

Avisi will be able to leverage these findings to implement a multi-cluster architecture within the AME platform and use it as a unique selling point to attract large enterprises as their new customers.

3 Avisi Managed Environments

The focus of this chapter is answering the first research sub-question – “What is the Avisi Managed Environments (AME) platform?”. Gaining in-depth understanding of the platform’s current situation and the architectural decisions made to build it is crucial before solving the problem laid out in this project. This is because the solution created during this project aims to extend the existing platform with new capabilities; it must integrate well within the current technological ecosystem of AME, not to hinder the functionality that is offered currently.

At first, this chapter talks about what AME is in general, as a product and service touching on who the platform customers are and what problems AME solves for them. It also talks about the advantages of the AME platform over other similar solutions such as Amazon EKS. Then, the chapter describes business context. It does not provide a full-fledged SWOT analysis, but it covers the most important aspects of the business context for this project: strengths and weaknesses within the team and the current architecture, opportunities and threats associated with pursuing a multi-cloud architecture. Finally, the chapter dives into the architecture of the platform. It looks at the key components and technologies, how Kubernetes is used, and which components are responsible for provisioning and managing customer environments in a fully automated manner. Furthermore, it describes which underlying infrastructure resources are deployed in AWS and other cloud providers to power the platform. Understanding all this context surrounding the problem is critically important to ensure that the future solution can seamlessly integrate within the current situation.

The primary output from this chapter are the **design constraints** formulated at the end. Design constraints can have either a business or technical nature and they play a critical role in designing the final solution as they narrow down the spectrum of possible solutions and ensure they fit within the current platform.

3.1 Methodology

Gaining information about the platform and its architecture will be done using the “Field” research strategy. The input in this research question are the records of the initial discussions of the assignment with stakeholders. For data collection, the primary methods will be Document Analysis, Interviews, Observation, and Literature Study.

To collect the general information about the AME platform, its customers, and the problems it solves, internal business and marketing documentation will be analyzed. After this, the CTO of Avisi, Jeroen Veldhorst, will be interviewed. He will be asked questions about the business, strategy and the vision for the platform, the opportunities and threats associated with this project. Since this project is technical in nature, a full-fledged SWOT analysis can be too heavy. However, certain aspects

The Document Analysis will be the starting point for understanding the architecture of the platform. Through analyzing the existing technical documentation of AME, I will be able to draw initial understanding of the architecture and potential constraints. Based on this, a list of questions will be prepared for the platform architect, Thomas Kooi and further information about the architecture will be derived from the interview with him. This is also important because some documentation might be incomplete or outdated.

Finally, to reinforce the understanding of the platform and see in real-time how it works and how people interact with it, I will request a walkthrough demo of the platform during which I will be able to observe the entire system in action. Additional literature study will be conducted to understand certain technical aspects of the current ecosystem of tools and technologies used at AME, for instance, to learn about the components of Kubernetes. This type of research will not be described in this report.

After collecting the information, it will be analyzed and processed into the design constraints. All derived constraints will be recorded and assigned a unique code for traceability. All design constraints, regardless of their origin, will be discussed with the platform architect to ensure they are valid.

3.2 AME – the foundation of Avisi Cloud

“Avisi Managed Environments” or AME is the foundational service offered by the Avisi Cloud unit. At its core, AME is a managed Platform as a Service (PaaS) solution that is based on the Kubernetes container-orchestration system which allows deploying, scaling, and managing containerized applications in a fully automated way. Like other managed container services such as Amazon EKS, AME exposes a web UI (console), CLI, and an API that allow organizations to create and manage fully dedicated environments where each environment consists of at least one Kubernetes cluster. However, AME has a few major advantages over other managed Kubernetes solutions like EKS.

The capabilities of the environments offered by AME stretch far beyond just running containerized workloads and keeping the control plane patched. One advantage is fully automated upgrades and patches of worker nodes (unlike EKS), this is achieved by using the paradigm of immutable infrastructure as explained in 3.4.3.2. Another significant advantage is out-of-the-box automation of operational services such as monitoring, logging, and alerting, tracing, disaster recovery, and policy enforcement – all of these services are at customers’ disposal as soon as the environment gets created. Besides, AME strives to achieve all of this in a provider-agnostic way – without relying on any specific offering from one cloud provider. Because while the Kubernetes part itself is cloud-agnostic, the rest of the platform such as Amazon EKS locks you in: log handling, monitoring, identity management, to name a few [2]. Whereas AME realizes all the core operational services by using industry-leading open-source solutions such as Prometheus and Cortex that can be the same regardless of the cloud provider. This allows the AME platform and its customers to drastically reduce vendor lock-in.

Additionally, AME provides the foundation for other important services developed and offered by Avisi Cloud such as Performance Engineering – a stack of tools that provides in-depth metrics and advice on the performance of applications that run in AME, thus offering the developers a clear view on how to improve their applications' performance.

To sum up, with Avisi Managed Environments customers receive an “all set” Kubernetes environment equipped with all the necessary tooling and support to run, scale, and manage their containerized applications. The platform is being developed and maintained by a team of the same name – team AME. AME takes charge of ensuring all environments are available, fast, resilient, and secure at all times allowing customers to host their business-critical applications and have fewer Site-Reliability-Engineers. Customers are free of any operational complexities and can fully focus on developing and improving their applications while saving costs on training and management for Kubernetes.

3.3 Business Context

This sub-chapter covers aspects of business context discovered through interviews and relevant to this project.

3.3.1 Multi-cluster – an opportunity to grow

Avisi wants to grow the platform and target large-scale enterprise customers with mission-critical applications. From interviews, both the CTO and the Platform Architect agree that the multi-cluster project is a great opportunity to attract new customers through a unique selling point. The incentives for the project are flexibility and control. Multi-cluster can empower enterprises to choose infrastructure providers and geographical locations freely, make it easy to comply with data regulations. Furthermore, multi-cluster opens the door to fully automated disaster recovery, exceptional resiliency, and high availability.

During the interview with the CTO, I determined that the goal of Avisi is to start providing a beta-test version of multi-cluster by the end of 2022. This is an important business constraint **(BC-1)** to consider when evaluating potential solutions because Kubernetes multi-cluster solutions are at an early stage of their development and only a few of them might be ready for beta-testing by the end of 2022.

3.3.2 Strengths and Weaknesses of the AME team

One of the biggest strengths of the AME team is strong programming skills. In the context of this project, it means that AME is capable of building advanced custom automation. In case this project discovers a high-quality multi-cluster connectivity solution that does not have automation OOTB, it can be solved by AME.

One of the weaknesses is the team size – it's a small team working on large roadmap. Therefore, a complex multi-cluster solution that will require a lot of teams' capacity to implement won't fit.

According to one of the platform developers, this project is “easy to make difficult”. The final solution should consider not only the architecture of AME but also the team composition, it should be adoptable not only by AME experts but also the end-users.

3.4 Architecture Overview

The architecture of AME is innovative and rather complex. To reflect the cloud-agnostic vision, many intriguing technological choices had to be made. Of course, this backstage complexity does not exist for the end-users – all they see is a normal Kubernetes cluster. However, for this project, it’s critically important to understand the current architecture to ensure the design developed during this graduation integrates seamlessly. This part of context analysis focuses on zooming into the primary aspects of architecture and identifying associated technical constraints.

This chapter begins with explaining what a container-orchestration system is, looking at Kubernetes in particular. After this, the concept of AME’s “Kubernetes-in-Kubernetes” is explained, followed by analyzing the underlying infrastructure resources and automation. This chapter does not provide an in-depth explanation of the whole architecture but rather focuses on specific components that are tightly related to this graduation assignment.

3.4.1 Kubernetes

Containers have revolutionized the way companies distribute their applications by offering exceptional portability, scalability, resource efficiency, isolation. Adoption of containers by companies was and is very rapid. According to [3], the median company that adopts Docker runs eight containers simultaneously on each host, a figure that has climbed steadily over the years. Due to this, organizations started facing a challenge: how to manage all the “cattle” effectively and efficiently? Container orchestration systems came in to solve these challenges by providing fully automated mechanisms to deploy, scale, and manage the entire lifecycle of containers from a central control plane. Today, containerized microservices are the foundation for cloud-native applications [4].

Kubernetes is one of the most widely adopted open-source container orchestration systems. In fact, Kubernetes has established itself as the de facto standard for container orchestration and is the flagship project of the Cloud Native Computing Foundation (CNCF), backed by key players like Google, AWS, Microsoft, IBM, Intel, Cisco, and Red Hat [5].

Kubernetes clusters are at the center of all AME internal and external services. AME uses its own Kubernetes distribution called AME Kubernetes. For the most part, it’s 100% “vanilla” Kubernetes except for a few adjustments. The main difference is that AME Kubernetes enforces default settings for higher security. Other differences mainly relate to the additional automation for operational services.

A few components of the core Kubernetes infrastructure are tightly related to this project. For instance, Container Network Interface (CNI) is responsible for inserting network interfaces into containers and thus providing connectivity of pods to networks (e.g., pod-to-pod, Internet). Since this project looks at cross-cluster networking, CNI will play a big role in the final solution. AME Kubernetes uses Calico CNI, and the final solution must be compatible with it **(TC-2)**.

AME strives to stay as close as possible to the upstream Kubernetes distribution as well as to follow the approaches fostered by the Cloud Native Computing Foundation (CNCF). In fact, Avisi has recently become an official member of the CNCF landscape. This is also expected to be followed in the Technical Design. **(BC-3)**

To summarize, several constraints can be drawn up at this point. The design must be compatible with the Calico CNI **(TC-2)**. Furthermore, it is desirable but not mandatory that the final solution uses components that are open-source **(BC-2)** and reside in the CNCF landscape **(BC-3)**.

3.4.2 Kubernetes-in-Kubernetes

The AME platform uses Kubernetes in a powerful and rather complex architecture called Kubernetes-in-Kubernetes. It is based around using an outer Kubernetes cluster to automate creation, upgrades and management of inner Kubernetes clusters used by end-customers. Subsequently, this architecture enables AME to provide customers with fully managed consistent clusters at scale.

The diagram below provides a high-level overview of this architecture.

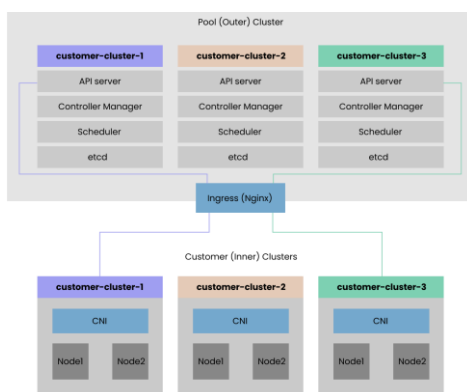


Figure 1. AME Kubernetes-in-Kubernetes architecture overview

The **pool** (outer) cluster runs control planes of multiple separate customer (inner) clusters. For each new customer, a namespace in the outer cluster gets created and then the control plane components are deployed as pods in that namespace. This ensures isolation and security. To provide high availability for the control planes, AME deploys several entities (pods) for every component.

A **customer cluster** has one or more node pools that run workloads – customer’s applications and data. Each node pool consists of one or more worker nodes (VMs) that provide resources like CPU, RAM, and storage. This is further explained in the next sub-chapter – Infrastructure.

Ingress controller (Nginx) is configured on the outer cluster to enable customer clusters to communicate with Kubernetes API server. The ingress controller is shared by all customers using the same outer cluster.

This architecture is very powerful but at the same time it adds complexity and constraints to this project. For example, if the future multi-cluster solution would require replacing certain components of the control plane or synchronizing multiple control planes, the solution might not integrate well. Therefore, it is a key technical constraint – the final solution must seamlessly integrate with AME Kubernetes **(TC-1)**.

3.4.3 Infrastructure

AME provides the PaaS platform but doesn't manage the underlying physical infrastructure such as hypervisors, load balancers or storage. The physical infrastructure is provided by external vendors.

3.4.3.1 Cloud providers

The AME platform is multi-cloud and provider-agnostic by vision, it uses independent components to be able to deploy anywhere. Currently, most of the platform runs on AWS but the platform is expanding support for more providers.

Most customer clusters worker nodes are AWS EC2 instances (VMs), storage is provided by Amazon EBS. Other providers are used for customer clusters' worker nodes are DigitalOcean, Scaleway, and Intermax.

During a discussion with the platform architect, it's been discussed that not all cloud providers support the same set of network protocols, and this is a technical constraint for cross-cluster connectivity. The final solution may only use network protocols supported by all providers mentioned above **(TC-4)**.

3.4.3.2 Immutable Infrastructure

Immutable infrastructure is an infrastructure paradigm in which servers are never modified after they're deployed [6]. This pattern is used by AME to automate the upgrades for Kubernetes worker nodes. Regular upgrades are crucial to keeping the system secure and stable, but the upgrade process can break applications running on the node. To prevent this, nodes are never patched or upgraded in place. Instead, new nodes get created with the desired versions and configurations. After they're validated, they're put into use and the old ones are decommissioned. This way, AME guarantees predictable upgrades without any unexpected configuration drifts. This is a technical constraint **(TC-3)** because the final solution cannot rely on changing node configuration in place.

3.4.4 Automation

The AME platform is designed and built with a strong focus on automation and a mindset that applications are not static – in fact, they are changing all the time, and the infrastructure must be consistent with this dynamicity to maintain excellent application availability and performance.

Each customer environment consists of 3 components: Kubernetes clusters, core infrastructure and operation services. All these components are automated following a GitOps approach. GitOps is a process of automating IT infrastructure using infrastructure as code and source version control systems [7]. AME holds cluster configuration and manifest files in code repositories. Then, whenever a configuration change occurs, a tool like FluxCD synchronizes the new configuration with the cluster driving it to reach the desired state.

Because automation is so important, one of the important demands for the design is that it must be 100% automatable using the AME approach and toolset so that it can scale. This concern will be covered in a separate user story and research question (#7).

3.5 Conclusion and Design Constraints

The architecture of AME is rather complex and unique. There are a few very distinct technical constraints related to networking that will have an impact on the technical design. Firstly, these technical constraints need to be studied more thoroughly through literature study. Secondly, to ensure they are considered, constraints will be included in the multi-criteria analysis alongside the user stories. Prototyping will be used as well to validate the constraints are met. The following design constraints have been drawn up.

BC – Business Constraint; **TC** – Technical Constraint

ID	Description
BC-1	Technologies used in the final solution must be ready for beta-testing by the end of 2022
BC-2	Technologies used in the final solution should be open source (preferable but not mandatory)
BC-3	Technologies used in the final solution should be CNCF-hosted (preferable but not mandatory)
TC-1	The final solution must seamlessly integrate with AME Kubernetes
TC-2	The final solution must be compatible with the Calico CNI (mandatory) and should be CNI-agnostic (preferable)
TC-3	The final solution should follow the pattern of immutable infrastructure
TC-4	Network protocols used in the final solution must be supported by the cloud providers used by AME: AWS, Intermax, Scaleway, DigitalOcean

Table 2. Design Constraints for AME multi-cluster solution

4 The Limits of the AME platform

The existing architecture of AME is incredibly powerful and already solves a lot of problems for the customers of Avisi. However, certain challenges related to jurisdictional compliance, disaster recovery, flexibility and other aspects of the platform remain unsolved. These challenges are becoming more relevant for the platform growth and for the type of customers that AME wants to target.

This chapter focuses on the second research sub-question: “Why does the current architecture limit AME in supporting more cloud providers and geographic locations?”. This research question unfolds the initial idea about the problem and gets to the core of it. This technical problem analysis will help to find root causes of the problem within the current architecture and address them in the Technical Design.

4.1 Methodology

Previously, the high-level architecture was analyzed. This research question will require zooming into certain parts of the architecture to get to the core of the problem. First, the existing design documentation will be examined with the focus on AME k8s control plane, networking, and storage. Further information about the problem from different perspectives will be derived from interviews with the CTO, Platform Architect, DevOps engineers, and internal customers of AME.

To process collected information and get to the origins of the problems and limitations within architecture, tools such as 5 Why's and Fishbone will be used. Both tools will focus on technical details of the architecture. Fishbone is used because there might be multiple technical root causes. In addition, prototyping will be used to better understand the limits of the current architecture through experiments.

4.2 Problem – Deep Dive

AME wants to grow the platform, expand its reach, and provide the customers with the sought-after capabilities described above. Since the AME platform is based on the container orchestration system – Kubernetes, proper cross-cluster networking will play a critical role in achieving these capabilities. That's why the topics of “multi-cluster connectivity” and “multi-cloud architecture” go toe to toe in this research and are both critically important for the goals of AME.

The previous chapter outlines the initial understanding of the problem – the current AME platform has limitations that need to be addressed to target enterprise customers. This chapter will deepen that understanding and get to the technical root cause that will become the primary focus of the Technical Design using 5 Whys. The diagram below summarizes the results from initial problem to

the root cause.



Figure 2. Analyzing the AME problem using 5 Why's

Why cannot a single Kubernetes cluster span multiple regions?

According to [8], it is not practical to stretch a single K8s cluster across geographically significant distances because of the etcd sensitivity to network latency and bandwidth between master nodes. Etcd is a key-value store component responsible for holding and managing critical data required by Kubernetes: state, configuration, metadata, API objects etc. A distributed protocol called Raft is used by etcd nodes to reach a consensus – in other words, a guarantee that all etcd nodes agree on a particular value. Even an insignificant increase in latency can cause timeouts resulting in a cluster's inability to start new workloads. Therefore, the master nodes of a single K8s cluster must reside within a single region.

4.3 Problem – Core

The **core of the problem** comes down to network **connectivity** between K8s clusters. To be able to quickly migrate workloads across regions and providers in response to jurisdiction changes or infrastructure disasters, or to have multi-region and multi-cloud deployments for enterprise-grade availability, pods on one cluster must be able to directly communicate with pods on other clusters in a multi-cluster topology. This connectivity must be reliable, fast, and secure. Achieving a multi-

cluster architecture with such connectivity will be the primary focus of the technical research and design.

The **secondary problem**, or rather challenge in a multi-cluster K8s architecture is **orchestration**. Orchestration within one K8s cluster is straightforward – the control plane takes care of everything automatically; this is what K8s is for. But when it comes to having a single point of orchestration for multiple clusters, issues arise. Orchestrating workloads across multiple clusters requires additional tooling and mechanisms that would allow control planes of those clusters to synchronize with each other.

After a discussion with the Platform Architect, it's been agreed that the **primary focus** of this project will be on **connectivity**. The orchestration challenge is of secondary priority in this project. Nonetheless, orchestration aspects will be addressed in the research and design granted that there will be sufficient time.

5 Stakeholders and User Stories

This chapter focuses on answering the research sub-question 4: “What requirements do the stakeholders at the company have for the future solution?”. It presents a condensed version of the Stakeholder Analysis including a brief take on the methodology, identified stakeholders (shortlist), summary of the interviews, and the resulting user stories. The full Stakeholder Analysis can be found in Appendix A.

5.1 Methodology

The primary objectives for this Stakeholder Analysis are to identify the people involved, understand their interests and influence in the project, gain insight into their wishes for the final product and obtain the necessary support and resources. The ultimate outcome of the Stakeholder Analysis is the list of User Stories for the final product.

For stakeholder identification, methods of Document Analysis (Avisi’s organizational chart) and Brainstorming will be used. During this assignment, I will not be in contact with external customers; instead, their interests will be represented by the internal customers of the platform (AME is also used internally) and the Platform Architect. The internal customers are the AME team itself as well as other teams in the Avisi Cloud unit.

First, a longlist will be compiled after which the Mendelow’s Power-Interest grid will be used to prioritize the stakeholders and establish a shortlist of people for requirements elicitation interviews. Not all requirements will be derived from interviews; due to the novel nature of the multi-cluster technology, some requirements will be sourced from literature study and community research. All requirements including business, functional and non-functional will be formulated as User Stories because this is a standard practice at AME. Further, the User Stories will be prioritized using MoSCoW. Due to a non-waterfall approach and experimental nature of the project, there will be multiple iterations of User Stories agreed with the Platform Architect.

For full methodology, see Appendix A (1.1 Methodology).

5.2 Stakeholders – Shortlist

This sub-chapter lists the stakeholders who possess the most interest, influence, and support in the project; they will be interviewed individually and consulted throughout the entire duration of the project. For the full list of stakeholders, see Appendix A (1.2 Stakeholders Full List)

ID	Stakeholder
INT-S1	Thomas Kooi, Product Owner, Platform Architect, Company Supervisor
INT-S2	Jeroen Veldhorst, CTO
INT-S3	Mark Freriks, Platform Developer
INT-S4	Bob Beeke, Platform Operations

INT-S5 Albert van de Kamp, Team Lead, Platform User (Internal Customer)

Table 3. Shortlist of stakeholders for requirements gathering

5.3 Summary of Stakeholder Interviews

This sub-chapter briefly discusses all interviews. A summary of each individual interview can be found in Appendix A.

At a high level, the stakeholders' wishes, and demands are in line: enabling network interconnection between different Kubernetes clusters, making it integrable with the current configuration of AME clusters, ensuring high security and maintainability support. Multi-cluster networking will enable AME to create Kubernetes environments that span multiple regions and clouds. It is crucial for the architecture to be reusable and well-fitted within the current ecosystem of the AME platform.

There have been some discrepancies among stakeholders on which use cases are the most relevant for Avisi Cloud and its customers. Nevertheless, the survey (Appendix A, Ch. 1.5) helped to establish the use cases baseline to match the User Stories. The proposed multi-cluster solution must enable building architectures (application and platform) to facilitate jurisdiction compliance, disaster recovery, high availability, latency reduction, and provider-specific features. It's precisely these use cases that Avisi can leverage to create a unique selling point for the AME platform and attract the type of customers Avisi wants to target – enterprises with large-scale mission-critical applications.

One of the critical points mentioned by all stakeholders is automation. Multi-cluster architecture may involve complex low-level networking solutions such as custom VPN tunnels and Direct Links. The solution should strive to make it easy for all types of users without special networking knowledge: developers, cluster operators, users; it should abstract and automate most of the multi-cluster connectivity. Other critical arguments are resiliency and security: introducing multi-cluster should not break any existing features of AME single-clusters and should minimize the security risks.

Interviews resulted in a list of user stories that represent the desired situation from different perspectives i.e., platform customers, developers, operators. Further, during the interview with the Platform Architect, it was discussed that multi-cluster networking should be the focus. Therefore, User Stories associated with the multi-cluster networking problem will have higher priority than the ones related to orchestration.

5.4 User Stories

The following user stories have been formulated and prioritized through interviews and review sessions with stakeholders. These user stories form the accepted scope accepted by the Product Owner. Each user story defines a set of acceptance criteria which will be used to form test cases and evaluate whether that user story is achieved at the end of the project.

User Story ID Construction

US – User Story; sequential number starting from 1; B – Business / F – Functional / NF – Non-Functional

5.4.1 Business

These User Stories represent the vision of Avisi Cloud for this project with the emphasis on the expected benefits for the business; they are derived primarily from the CTO and Platform Architect.

ID	Description	Priority	Acceptance Criteria
US-1-B	As a CTO, I want to make the AME platform more flexible through multi-cluster Kubernetes architecture so that I can create a unique selling point to attract new customers.	MUST	<ul style="list-style-type: none">• Sign-off from the CTO
US-2-B	As a Platform Architect, I want to have a multi-cluster solution that does not rely on any specific implementation offered by one cloud provider, so that the platform remains provider-agnostic.	MUST	<ul style="list-style-type: none">• The final design can be deployed on any provider• Sign-off from the Platform Architect
US-3-B	As a Platform Architect, I want the final solution to scale to 100 clusters without hindering my team with manual activities and maintenance.	MUST	<ul style="list-style-type: none">• Manual activities are demonstrated• Manual configuration should take less than 5 minutes
US-4-B	As a Platform Architect, I want the Advisory Report to include information on how Avisi can use multi-cluster topology for the GAIA-X project so that it can be used for platform growth in the future.	COULD	<ul style="list-style-type: none">• The final report contains advice on the use of multi-cluster in relation to the GAIA-X initiative

Table 4. Business-related User Stories

5.4.2 Functional

Functional User Stories focus on users' needs around the functionality of the multi-cluster solution, i.e., what actions users should be able to perform in the desired architecture. These user stories are mostly derived from the Platform Architect as he represents customers' interests but also from the stakeholders who work on the platform – developers and operators.

ID	Description	Priority	Acceptance Criteria
US-5-F	As a Platform User, I want to be able to fully replicate a service across multiple clusters in a multi-cluster topology so that I can achieve higher availability, resiliency, and reduced latency for my mission-critical applications.	MUST	<ul style="list-style-type: none">• Demonstrate a mock application deployed across 2 clusters (fully replicated)• Demonstrate HA and resiliency by destroying one cluster; the application should still be available• Demonstrate latency by accessing the mock application from different regions

US-6-F	As a Platform User, I want to be able to segment a service across multiple clusters in a multi-cluster topology so that I can effectively deal with jurisdiction compliance.	MUST	<ul style="list-style-type: none"> Demonstrate separate services of a mock application deployed across 2 clusters; different services can communicate with each other directly
US-7-F	As a Platform User, I want to be able to migrate an application from cluster to cluster in a multi-cluster topology transparently so that I can effectively respond to various changes within the environment.	SHOULD	<ul style="list-style-type: none"> Demonstrate the migration of a mock application from one cluster to another cluster with minimum downtime
US-8-F	As a Platform User, I want the multi-cluster environment to support soft multi-tenancy so that different departments or teams within my organization can share the environment.	SHOULD	<ul style="list-style-type: none"> Demonstrate the means of creating soft multi-tenancy across 2 clusters (for example, replication of namespaces)
US-9-F	As a Platform User, I want to be able to firewall traffic across interconnected clusters so that I can control which pods can communicate with each other.	MUST	<ul style="list-style-type: none"> Demonstrate the means of controlling network traffic flow at the IP address or port level across 2 clusters.
US-10-F	As a Platform User, I want to be able to prioritize traffic to the local cluster over a remote cluster so that bandwidth costs for the traffic that crosses the cluster boundaries can be minimized.	SHOULD	<ul style="list-style-type: none"> Demonstrate the mechanisms of enforcing a local cluster traffic before routing the traffic to a remote cluster
US-11-F	As a Platform Architect, I want to have encryption mechanisms for cross-cluster traffic so that the security of data can be ensured.	MUST	<ul style="list-style-type: none"> Demonstrate that all cross-cluster traffic flows through a secure tunnel connection between two clusters
US-12-F	As a Platform Ops, I want to be able to use the existing toolset with the multi-cluster solution so that I can easily monitor, troubleshoot, and debug the solution.	SHOULD	<ul style="list-style-type: none"> Demonstrate that metrics and logs for the multi-cluster topology can be collected using the existing AME toolset (Prometheus, Grafana, Loki)
US-13-F	As a Platform User, I want to be able to schedule pods on a specific cluster in the topology so that I can fulfill advanced scheduling requirement.	SHOULD	<ul style="list-style-type: none"> Demonstrate the ability to assign a pod to run (or not run) on a certain cluster among the 2 interconnected clusters
US-14-F	As Platform Ops, I want to have a runbook for the final solution so that I know which steps need to be performed to configure and maintain a multi-cluster topology.	MUST	<ul style="list-style-type: none"> The runbook has been recorded in AME documentation on Confluence The runbook contains a description, step-by-step procedure (commands), and references to documentation The runbook has been tested and approved by an ops engineer

Table 5. Functional User Stories

5.4.3 Non-Functional

The User Stories in this sub-chapter focus on non-functional requirements; they address how the future multi-cluster topology must behave and perform in terms of security, speed, usability,

reliability, etc. These user stories have been sources primarily from literature study and community research.

ID	Description	Priority	Acceptance Criteria
US-15-NF	As a Platform Architect, I want the final solution to be aligned with the security principles of AME so that the multi-cluster topology does not create serious security vulnerabilities.	MUST	<ul style="list-style-type: none"> The documentation describes how the solution adheres to the security principles Sign-off from the Product Owner
US-16-NF	As a Platform User, I want to be able to connect clusters within a few clicks using the AME console, so that I don't have to perform any complex configurations.	WON'T	Out of scope
US-17-NF	As a Platform Architect, I want the network overhead (latency, throughput, CPU usage) in the multi-cluster topology to be less than 10% (relative to a single cluster) so that the impact on user experience can be minimized.	SHOULD	<ul style="list-style-type: none"> Network stats are tested and recorded by accessing a mock application deployed on a single cluster Network stats are tested for the same application deployed across two interconnected clusters (in the same region) The difference between the stats is not higher than 10%
US-18-NF	As a Platform Architect, I want the multi-cluster solution to permit automated pod-to-pod service discovery, load balancing, and routing for multi-cluster services so that I don't have to make any additional configurations.	MUST	<ul style="list-style-type: none"> Demonstrate an ability to resolve hostnames from a component of a cluster across clusters
US-19-NF	As a Platform Architect, I want the multi-cluster solution to be fully automatable so that I don't have to perform any repetitive manual configurations for each cluster.	MUST	<ul style="list-style-type: none"> The final report explains how the multi-cluster interconnection setup and deployments can be automated
US-20-NF	As a Platform Ops, I want the solution to have resiliency mechanisms so that failure in one cluster does not cause the entire multi-cluster topology to fail.	SHOULD	<ul style="list-style-type: none"> Demonstrate an environment with three interconnected clusters in which upon a destruction of one of the clusters, the multi-cluster connectivity remains functional
US-21-NF	As a Platform User, I want to be able to interconnect more than two clusters so that I can deploy my application in at least 3 regions.	COULD	<ul style="list-style-type: none"> Demonstrate the interconnection of 3 clusters in different regions
US-22-NF	As a Platform User, I want the solution to have support for overlapping pod IP addresses so that there are no routing issues across interconnected clusters.	MUST	<ul style="list-style-type: none"> Demonstrate the absence of routing conflicts when interconnecting clusters with overlapping Pod, Service and Cluster subnet CIDRs

Table 6. Non-functional User Stories

6 Technical Research: K8s Multi-cluster

This chapter gathers all technical research conducted throughout the project. It includes research about best-practices and solutions for K8s multi-cluster to answer the research question 4. Further, it talks about automation techniques to answer the research questions 7, as well as challenges and risks of the architecture to answer the research question 9.

6.1 Methodology

Research sub-questions 4, 7 and 9 will be broken down into spikes – time-boxed investigations of a technical topic required to fulfil a user story. Spikes are aligned with user stories; they are specific and allow to conduct targeted technical research.

The table below represents the research spikes based on story mapping, related user stories and sub-chapters in which the spikes will be elaborated.

Spike Description	Related User Stories	Refs
<i>Multi-cloud K8s</i> What do provider-agnostic and multi-cloud strategies mean? How can these strategies be achieved with K8s? Which reference architectures exist for multi-cluster?	US-1-B, US-2-B	6.2; 7.2;
<i>Connectivity</i> How does networking work in K8s? Why is it challenging to connect clusters? What are the best technologies to solve the connectivity problem? How to evaluate network overhead? How to control traffic across clusters (firewall and prioritization)? How to ensure service discovery? How to ensure fault-tolerance in the topology? How to ensure support for overlapping IP CIDRs?	US-9-F, US-10-F, US-13-F, US-17-NF, US-18-NF, US-20-NF, US-22-NF	6.3.1; 7.1.3; Appx B: 2.3.2, 2.4.2, 2.5.2, 2.6.2
<i>Security</i> How to realize encryption for cross-cluster communication? How to adhere to security principles of AME?	US-11-F, US-15-NF	Appx B: 2.3.3, 2.4.3, 2.5.3, 2.6.3
<i>Maintainability</i> How to integrate with existing monitoring and logging tools? How to automate multi-cluster connectivity?	US-3-B, US-12-F, US-14-F, US-19-NF	Appx B: 2.3.4, 2.4.4, 2.5.4, 2.6.4
<i>Deployments and Orchestration</i> How can soft-multitenancy be achieved in a multi-cluster environment?	US-5-F, US-6-F, US-7-F,	10.1.1; 10.1.2; 10.1.3

How to schedule pods on specific clusters in the topology?	US-8-F
How to deploy applications and data to multi-cluster?	
How to migrate applications in multi-cluster?	
How to deploy a global load balancer and storage in multi-cluster?	

Table 7. Research Spikes

Finding information during the spikes will be done primarily through Library strategy. A literature study will be performed to analyze state-of-the-art publications and conferences related to K8s multi-cluster architectures as well as CNCF sources. Further information will be collected through community research (K8s Multi-Cluster Special Interest Group, GitHub, etc.).

First, a longlist of multi-cluster connectivity projects will be formed. Then, the projects will be checked against the constraints to eliminate the options that cannot be used within AME; this will result in a shortlist. After that, shortlisted projects will be evaluated against AME requirements to determine the best-fit solution; for this, multi-criteria analysis will be used along with prototyping.

6.2 Provider-agnostic and Multi-cloud K8s

The term of provider-agnostic (or vendor-agnostic) refers to tools, platforms and applications that are compatible with any cloud infrastructure. They can be moved to and from different cloud (and on-prem) environments freely without inducing any operational issues. A business is cloud-agnostic when the company IT systems are not locked into a single cloud vendor or do not rely on one cloud provider's proprietary services [9].

Kubernetes is provider-agnostic by design, but only the open-source, "vanilla" distribution – it can be deployed in any datacenter or cloud if all infrastructure preconditions are satisfied. However, most public cloud-based distributions of K8s like EKS are not vendor-agnostic as they induce lock-in through integrations with vendor-specific services and tools on top of K8s, for example, with AWS eksctl [10]. The AME Kubernetes minimizes the use of vendor-specific resources in pursuit of maintaining the platform vendor-agnostic.

The terms "provider-agnostic" and "multi-cloud" are easy to confuse but they also go together in today's cloud strategies. While provider-agnostic means that a company is not locked-in within a single vendor, multi-cloud means that it utilizes several clouds at the same time. For instance, in a hybrid cloud scenario, a company might combine resources from their own private cloud with resources from a public cloud like Azure. Using multiple public clouds within one organization is also not uncommon. Multi-cloud is a strategy pursued by organizations to become vendor-agnostic.

Multi-cloud Kubernetes is a big challenge because a K8s cluster is siloed within a single region and cloud. Kubernetes is not designed to support a single cluster that spans multiple providers or regions on the wide area network [11]. To build a multi-cloud or multi-region Kubernetes environment there

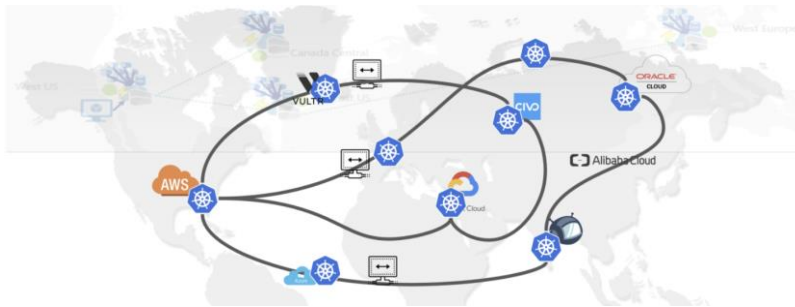


Figure 3. K8s Multi-cloud and multi-region distribution; adapted from [49]

must be multiple K8s clusters interconnected in a multi-cluster topology; they should be able to directly communicate and share resources with each other. Hence, in this research, the terms “multi-cloud” and “multi-cluster” are interchangeable.

Multi-cluster topologies introduce two classes of challenges: *synchronization*

between cluster control planes and *interconnection* that makes services directly accessible across clusters [12]. This research aims to focus on the interconnection challenge by evaluating various connectivity solutions. The next sub-chapter dives deeper into how networking is realized in K8s and what is the best way to connect multiple clusters.

6.3 K8s Networking and Multi-cluster Connectivity

6.3.1 Overview of K8s Networking

Networking is a central part of Kubernetes, it allows different K8s components to communicate with each other and with other applications. K8s networking has 4 distinct areas of communication: container-to-container (within a Pod), Pod-to-Pod, Pod-to-Service, External-to-Service [13].

K8s provides the networking model to enable the above connections but the implementation of that model is not provided out-of-the-box. The most common way to realize the networking in K8s is through Container Network Interfaces (CNI) plugins. A CNI plugin is responsible for inserting a network interface (veth – Virtual Ethernet) into the container network namespace and making the necessary changes on the host and in the routes [14]. One of the foundational principles of K8s networking is that Pods can communicate with each other directly in a flat network across all nodes using their own IP addresses. This behavior must be preserved in a multi-cluster topology.

Current official K8s documentation counts at least 25 different implementations of the K8s networking model – each having details that differ from each other. This is a challenge for multi-cluster scenarios that this research is focused on. There is a need to communicate across separate private networks residing in different regions and clouds reliably and securely, however, most networking solutions are not providing any guarantee about how multi-cluster works making it completely left up to the unintentional implementation details [11]. And although today all AME clusters use Calico as a CNI, it should not be assumed that this will not change in the future.

Finding a reliable, secure, and flexible inter-cluster networking solution that respects all constraints and requirements of AME is crucial. The next sub-chapters look at various projects and solutions.

6.3.2 Multi-cluster Connectivity Solutions – Longlist

As explained in 6.3.1, CNIs provide networking in a single cluster. A multi-cluster connectivity solution should enable direct cross-cluster pod-to-pod and pod-to-service communication required for a multi-cluster topology. A literature study of the CNFC sources showed that this can be achieved either through CNI extensions, or by dedicated tools [12]. Multi-cluster connectivity solutions vary in their designs, where critical design choices involve mainly three aspects: interoperability with different cluster configurations, compatibility between network parameters used in the other clusters, and dealing with services exposed on all clusters.

A CNCF literature study combined with K8s community research resulted in a list of connectivity solutions that are being actively worked on today; all of them are open source. Proprietary solutions such as AWS Outposts, Google Anthos are excluded from the evaluation due to provider-agnostic principle.

Below is the full list of projects discovered in this research. Generally, the solutions can be divided into two groups: VPN-based and Gateway-/Proxy-based; both having advantages and disadvantages that will be discussed further.

Project name	Description	CNCF Maturity (as of Jan 26, 2022)
Submariner	Submariner enables direct networking between Pods and Services in different K8s clusters, either on-premises or in the cloud through L3 connectivity using encrypted Wireguard or IPSec VPN tunnels [15].	Sandbox
Skupper	Skupper is a layer 7 service interconnect. It enables secure communication across Kubernetes clusters with no VPNs or special firewall rules using HTTP/1.1, HTTP/2, gRPC, and TCP communication secured by mTLS [16].	Not listed
Liqo	Liqo aims to create a CNI-agnostic and straightforward way to create decentralized multi-cluster topologies using Wireguard tunnels. On top of connectivity, Liqo proposes a seamless multi-cluster model which allows Pods and Services to be offloaded on remote clusters [17].	Not listed
Cilium ClusterMesh	Cilium's multi-cluster implementation that provides Pod IP routing across multiple K8s clusters at native performance via tunneling or direct routing without requiring any gateways or proxies. It uses transparent encryption for all communication across cluster boundaries [18].	Incubating
Istio Multi-cluster	Istio is a service mesh with multi-cluster support. The interconnection among different clusters uses a dedicated proxy	Not listed

	to route traffic from the mesh of one cluster to another. Traffic is secured by mTLS [19].	
Linkerd Multi-cluster	Linkerd is a service mesh. Its multi-cluster support works by “mirroring” service information between clusters. Connection between clusters is encrypted and authenticated on both sides with mTLS [20].	Graduated
NSM	Network Service Mesh (NSM) is the Hybrid/Multi-cloud IP Service Mesh enabling zero-trust L3 between individual K8s Pods across clusters/clouds. Can be combined with L7 service meshes and is not limited to K8s [21].	Sandbox
kEdge	kEdge is a proxy for gRPC and HTTP microservices that aims to make cross-cluster microservice communication secure and simple to set up. Uses TLS and special dialer to enable pod-to-pod [22].	Not listed
Consul Mesh Gateway	Consul is a service networking solution that can enable multi-cluster WAN federation via Mesh Gateways. It can be used to federate K8s clusters across cloud providers with end-to-end mTLS encryption [23].	Not listed

Table 8. Overview of active multi-cluster connectivity projects

Some projects like Submariner are dedicated specifically to enabling direct communication between pods and services in different K8s clusters while others like NSM can achieve the same result but require much more complex configuration due to low-level networking. Further, some solutions like Cilium ClusterMesh have major drawbacks, for instance, introduce strict dependencies on specific CNIs or assume a certain configuration of a cluster’s network, e.g., require unique Pod CIDRs in each cluster. Due to this, not all projects can be considered for AME. First, the solutions listed above will be filtered using the constraints determined in 3.5. Projects that do not follow the mandatory AME constraints will be excluded from the multi-criteria analysis.

Table 9 condenses the relationships between the discovered connectivity projects and the AME constraints. The plus (+) sign means that a solution fully conforms with a constraint, the minus (–) sign means that it does not, and the plus/minus (+/–) implies that a solution conforms to a constraint partially. The solutions colored green will proceed to the shortlist, whereas the red ones will be excluded. A detailed explanation per solution can be found in Appendix B, chapter 2.1.

Constraint	Subm.	Skup.	Liqo	Cil.	Ist.	Link.	NSM	kEdge	Cons.
BC-1	+	+	+	+	+	+	+	+	+
BC-2	+	+	+	+	+	+	+	+	+
BC-3	+	–	–	+	–	+	+	–	–
TC-1	+	+	+	–	+/–	+	+/–	+	+
TC-2	+/–	+	+/–	–	+	+	+	+	+
TC-3	+	+	+	+	+	+	+	+	+
TC-4	+	+	+	+	+	+	+	+	+

Table 9. Constraints Evaluation of Multi-cluster Solutions

6.3.2.1 Conclusion

Nine different multi-cluster connectivity projects have been discovered and evaluated against AME constraints. All discovered solutions are ready for beta-testing, and some have already gone through extensive testing in production clusters by community users. All discovered solutions are open-source, however, only four out of nine solutions are CNCF-hosted projects.

Five projects will be excluded from further evaluation, namely, Cilium ClusterMesh, Skupper, Istio Multi-Cluster, kEdge, Consul Mesh Gateway. Cilium ClusterMesh is excluded due to a strict dependency on Cilium CNI (does not conform with TC-2). Skupper, Istio Multi-cluster, kEdge and Consul Mesh Gateway will be eliminated for two reasons. First, they are not CNCF-hosted which (BC-3 non-conformant), however, the CNCF constraint is not mandatory. The second is explained in detail in Appendix C, chapter 2.1. In short, the projects in question are all proxy/gateway-based solutions that operate on Layer 7 and employ a very identical approach but at the same time, they are significantly less mature than Linkerd which, in its turn, has a CNCF-graduated maturity since 2017. There is little value in fully evaluating all of them. Therefore, it's been decided that out of all proxy-based solutions, only Linkerd will be considered. This decision has been approved by the Platform Architect.

One exception is Lipo. Despite not being on CNCF, the project has a lot of potential and attention in the community. Besides, Lipo has unique features that could be extremely valuable for AME. For this reason, the decision is to include Lipo in the shortlist.

6.3.3 Connectivity Solutions – Shortlist

After eliminating the connectivity solutions that do not comply with the constraints of AME, the following shortlist has been formed:

- Submariner – CNCF-sandboxed, cross-cluster L3 overlay network with broker-based topology
- Lipo – non-CNCF, cross-cluster L3 overlay network with peer-to-peer topology
- Linkerd Multi-cluster – CNCF-graduated, L7 service mesh with gateway-based cross-cluster connectivity
- NSM – CNCF-sandboxed, L3 zero-trust network between individual workloads/pods

The projects above will undergo extensive multi-criteria analysis against AME user stories. Based on the MCA, a decision will be made on which solutions will proceed to Technical Design.

6.3.4 Multi-criteria Analysis of Shortlisted Connectivity Solutions

This chapter focuses on a qualitative multi-criteria analysis of K8s multi-cluster connectivity solutions. The goal of this MCA is to establish which solution out of the shortlist qualifies the most in terms of the functional and non-functional requirements of AME. User stories defined in 5.4 will be used as criteria. Because this MCA is focused only on connectivity, user stories related to orchestration will be excluded from criteria.

Each solution will be evaluated against each criterion based on information gathered through available documentation, community posts, and prototyping, and assigned a score between 1 and 4 (explained in Table 10).

Score	Explanation
0	Not capable The evaluated product cannot satisfy the evaluated criteria
1	Capable with significant limitations The product can partially satisfy the criteria with certain non-critical limitations
2	Capable without significant limitations The product fully satisfies the criteria
3	Highly capable The product fully satisfies the criteria and stands out through additional features or automation

Table 10. MCA scores explanation

Criteria have different weights – one criterion might be more important than the other. Weights in this MCA will be based on the MoSCoW priorities defined for each user story (Table 11).

Priority	Weight	
Must	10	The final score is calculated by multiplying a score gained on a criterion by its weight. The solution that scores the highest will be used in Technical Design. Extended version of multi-criteria analysis demonstrates the scores for each criterion and product including the elaboration of assigned score, it can be found in Appendix C.
Should	5	
Could	2	
Won't	1	

Table 11. MCA criteria weight

6.3.5 Multi-criteria Analysis – Results

This sub-chapter discusses the total scores and associated conclusion of the MCA. For a complete analysis, refer to Appendix C.





			
Linkerd	Liqo	NSM	Submariner
259	211	199	179

Table 12. Multi-cluster solutions MCA – total scores

Table 12 shows the total scores received by the four evaluated technologies.

Linkerd scored the highest – 259. The MCA revealed that Linkerd is the only solution that satisfies all AME requirements and is rich with additional features beneficial for a multi-cluster environment such as distributed tracing and traffic split. Besides the core capability – L7 connectivity through multi-cluster gateways, Linkerd's strongest points are exceptional simplicity, security-first design with mTLS, traffic control with Authorization and Client-side (coming in 2022) policies. Linkerd has a few non-

critical drawbacks, e.g., with sidecar proxies, the CPU and latency overhead is inevitable, however Linkerd strives to reduce it to an absolute minimum. Overall, the MCA proves that Linkerd is simplest solution for secure Kubernetes multi-cluster connectivity as of today; it is suitable for a wide range of generic use cases for multi-cluster services that communicate over HTTP, gRPC and other TCP-based protocols as described by **DP-1** in 6.4.

Liqo's total score is the second highest – 211. Liqo is a unique, one-of-a-kind technology that addresses both multi-cluster challenges – networking and orchestration – in one solution. Liqo provides Layer 3 VPN-tunnel-based connectivity across clusters which is best-suited for when services need direct IP reachability as described by **DP-1** in 6.4. It also can seamlessly schedule workloads across connected clusters making it potentially the best solution for jurisdiction compliance and disaster recovery, application migration use cases. In addition, Liqo conforms all design principles established in 6.4, for example, it supports decentralized and dynamic topologies, does not make assumptions about a cluster configuration (CNI, CIDRs), fully based on vanilla K8s components and concepts. Currently, Liqo cannot address certain “MUST” requirements of AME, associated with traffic control (NetworkPolicy, local traffic priorities), however, according to the roadmap, this support will be introduced in the upcoming versions. All factors considered, Liqo will be proposed as one of the multi-cluster architectures in Technical Design.

NSM received 199 points in total. It does not satisfy all AME requirements and is rather complex. However, NSM has a few strong characteristics not presented by any other solution: the granularity of connections – NSM connects individual workloads instead of connecting entire clusters, full independence of a runtime domain (not limited to K8s) and internal K8s networking (CNI). NSM is not a generic solution; it is best-suited for a number of very distinct use cases: 1. establishing a vL3 between two individual K8s workloads in different clusters, for example, for replication between two database instances; 2. connecting an individual K8s workload to a non-K8s workload, for example, migrating a database from a VM in a datacenter to a Pod in a K8s cluster; 3. connecting an individual K8s workload to one or more external Network Services such as firewall or IPS. NSM is a lot more complex and involves low-level networking, therefore it cannot be used as a generic solution like Linkerd. Nonetheless, NSM-based architecture will be proposed to AME to address the specific use cases mentioned above.

Submariner, at last, received the lowest score of 179. This is due to Submariner being significantly limited both in terms of its features and architecture. Feature-wise, its weakest points are limited multi-cluster services (does not support having Endpoints spread across multiple clusters), no multi-cluster Network Policy or another mechanism to control traffic. Architecture-wise, Submariner contradicts certain design principles e.g., **DP-6** because Submariner depends on a central Broker shared by all clusters, and **DP-4** because Submariner expects clusters to be architected with non-overlapping CIDRs by default. Prototyping during the MCA also showed that Submariner is not stable. Considering all of the above, Submariner will be excluded from the final design proposals as it will not bring any additional value to the use cases of AME.

Based on the results of the MCA, the three leading solutions cover all cases described by DP-1 in 6.4. Therefore, the final Technical Design will include three architecture proposals:

- An architecture based on Linkerd for Layer 7 connectivity through L7 proxies and Multi-cluster Gateways.
- An architecture based on Liqo for Layer 3 connectivity through VPN tunnels.
- An architecture based on NSM for L2/L3 connectivity and non-standard protocols that use Ethernet/IP payloads.

6.4 Design Principles

Through spikes of technical research, it became evident that, at present, there is not a standard set of principles for multi-cluster solutions and topologies. Multi-clusters are still very new; neither networking nor orchestration across multiple K8s clusters has been standardized yet. Different projects define their own sets of design principles that fit their vision. Overall, many projects, especially the CNCF-hosted, share the same general direction and principles.

By analyzing a wide range of multi-cluster projects and their designs and setting them side by side with AME constraints and requirements, I've been able to accumulate a set of design principles that will act as a guiding compass for the future Technical Design of multi-clusters on AME.

ID	Description
DP-1	<p>Choose a connectivity layer based on application needs</p> <p>Multi-cluster connectivity can be established at different layers of OSI (L2-L7); the choice should be based on the applications' needs.</p> <ul style="list-style-type: none">• If services communicate over HTTP, gRPC, or other TCP-based protocols, L7 connectivity is optimal (e.g., L7 proxy, Service Mesh).• If applications require direct IP reachability, multi-cluster connectivity should be established at L3 (e.g., VPN Tunnels).• If workloads require lower-level networking features or non-standard protocols such as proprietary DB replication protocols based on Ethernet/IP payloads, an architecture based on L2/L3 networking should be chosen (e.g., Network Service Mesh).
DP-2	<p>Adopt a zero-trust security model</p> <p>Communication between clusters across regions and cloud providers entails cross-cluster traffic over the Internet exposing it to attackers. Further, multiple clusters create a larger attack surface than a single cluster environment. Therefore, it is essential that multi-cluster solution implements the necessary configurations to adopt the zero-trust networking by default (e.g., traffic encryption, explicit allow-rules etc.) This DP ensures the design will be aligned with AME "Security by design" principles.</p>
DP-3	<p>Stay as close to "vanilla" Kubernetes as possible</p> <p>The design should work with the native Kubernetes components and networking model. It should not create any disruptions for end-users in the way they interact with the cluster and deploy their applications. Most importantly, it shouldn't require any changes from within applications to enable multi-cluster connectivity. APIs should stay transparent.</p>

	The solution should use K8s concepts and patterns correctly. For example, leverage CRDs to extend the native API instead of abusing ConfigMaps.
DP-4	<p>Do not make assumptions about cluster configuration</p> <p>The design should not make any assumptions about specific configurations of a cluster (CNI, CIDRs). It should be possible to interconnect any CNCF-conformant K8s clusters that realize the standard K8s networking model. [24]</p> <p>This DP ensures, for instance, connecting clusters with overlapping Pod or Service CIDRs; or heterogenous networking scenarios (connect clusters with different CNIs)</p>
DP-5	<p>Keep it simple and automated</p> <p>The topology should be simple to set up, operate, maintain, and debug. The components should be well-defined and clear. The behavior should be understandable and predictable with no hidden “magic” involved. End-users should be able to start using the architecture with minimum training required.</p> <p>Connectivity aspects should be automated as much as possible (interconnection set-up, discovery, routing, etc.). The topology should be configured for auto-recovery. This DP will make sure that the topology can scale without operational overhead.</p>
DP-6	<p>Keep clusters independent and decentralize the topology</p> <p>Multi-cluster topology should preserve and respect failure domains. A topology with a “leading/central” cluster that all other participating clusters connect to should be avoided. Failures in one participating cluster should never impact other clusters. All participating clusters should be able to live on their own. If a centralized topology cannot be avoided, additional resiliency measures should be taken for the “primary” cluster.</p> <p>This DP ensures the topology is resilient to failure and draws the boundary between inter- and intra- cluster networking.</p>
DP-7	<p>Minimize network overhead and resource consumption</p> <p>First, the efficiency of communications in a multi-cluster topology should be kept as close to native network speeds as possible to mitigate negative impact on user experience. The traffic between Pods and Services in the topology should follow the order: 1. Local Node, 2. Local Cluster, 3. Remote Cluster, unless otherwise required.</p> <p>Second, the components of the topology running inside the cluster (e.g., multi-cluster connectivity agents or gateways) should consume the lowest possible amount of memory and CPU and be able to scale gracefully.</p>

Table 13. Design principles for a K8s multi-cluster architecture on AME

7 Conceptual Design

Before delving into technical design of the desired architecture, it is useful to understand the underlying key ideas and concepts of Kubernetes, the capabilities within a single cluster, and how they would extrapolate to multi-clusters. This chapter also looks at conceptual architectures for multi-cluster use cases. The focus is kept primarily on connectivity; however, orchestration aspects are also touched.

7.1 Kubernetes Concepts

Kubernetes is an industry-standard container orchestration system; it is the foundation of AME. A single self-sufficient Kubernetes environment is a cluster.

7.1.1 Cluster

When Kubernetes is deployed, the resulting entity is a cluster. A K8s cluster is a set of worker machines called nodes (usually, VMs) running containerized applications. The integral components that make up Kubernetes are control plane and worker nodes.

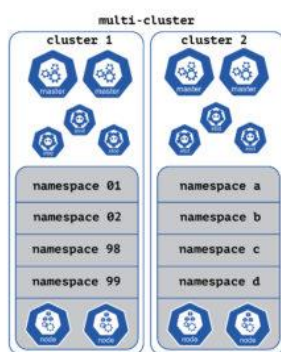


Figure 4. Multi-cluster with separate masters and workers; adapted from [25]

The control plane is a set of processes that control worker nodes including task assignment, state management, responding to changes in the cluster. It drives the actual state of the system to match the desired state.

Worker nodes are the data plane of a K8s cluster; they provide capacity such as CPU, memory, network, and storage to run containers. Every node has an agent called kubelet that ensures that desired containers are running and healthy.

In a multi-cluster environment, despite clusters being connected, they do not share any integral components. Each cluster has a distinct control plane and worker nodes; this way, participating clusters remain independent.

7.1.2 Containers, Pods and Workloads

Containers are the current cloud-native standard for application deployment. They encapsulate an application and all its dependencies into a single deployable unit, making deployments predictable and repeatable regardless of the underlying infrastructure or OS. Kubernetes brings the advantages of containers to the next level through Pods and workload resources.

Deploying applications to Kubernetes is a declarative process. A cluster operator specifies the desired state of a deployment including all parameters such as an image to be used by a container, number of replicas, etc.; then, K8s creates the necessary Pods. A Pod is the smallest and simplest Kubernetes object which represents a set of running containers and has a defined lifecycle fully managed by cluster's workload resources (controllers) such as Deployment, ReplicaSet, StatefulSet, DaemonSet, or Job. These controllers run in a continuous loop to ensure the right number and the right kind of Pods are running at any given time with respect to the specified desired state.

Scheduling means matching Pods to worker nodes for resource utilization. In a single cluster, the kube-scheduler takes care of assigning Pods to worker nodes. Orchestration is a broad term that covers automation of most operations involved into managing containers including scheduling, scaling, connecting and more. Kubernetes is designed to orchestrate containers within a single cluster but has no defined standard to do it across clusters. Multi-cluster orchestration can be achieved through a form of synchronized multi-cluster control plane – a single point for orchestration for multiple clusters. Further explained in 7.3.

7.1.3 Networking, Services and Load Balancing

The Kubernetes networking model imposes several fundamental requirements on any networking implementation as explained in [25]:

- every Pod gets its own IP address
- Pods on a node can communicate with all pods on all nodes without NAT
- Agents on a node (e.g., kubelet) can communicate with all Pods on that node

This creates a clean, backwards-compatible model where Pods can be treated much like VMs.

7.1.3.1 Services and Load Balancing

In K8s, Pods are considered nonpermanent – they get created and destroyed frequently to match the desired state of a cluster. Usually, different pods need to communicate with each other, but due to their ephemerality, this communication cannot rely on Pod IP addresses. K8s solves this problem using an abstraction resource called Service which defines a logical set of Pods and a policy by which to access them.

Further, by default, any workload launched in a cluster is only accessible internally. The Service resource provides a way to expose an application running in Pods to be reachable from outside the cluster.

Kubernetes provides multiple Service types with different levels of access:

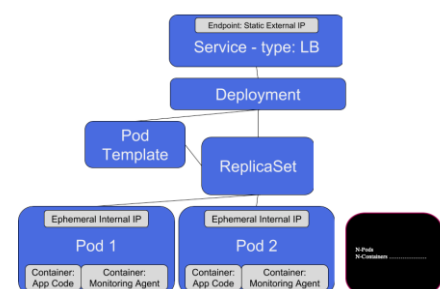


Figure 5. Kubernetes LoadBalancer Service; from [28]

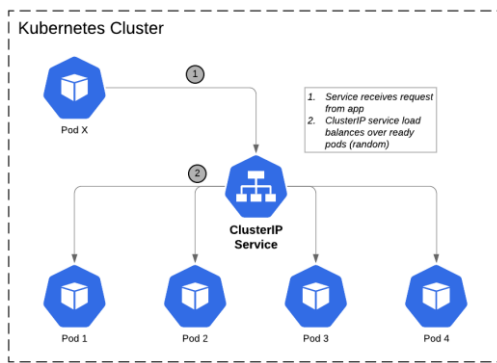


Figure 6. Kubernetes ClusterIP Service; from [27]

- “ClusterIP” which makes a service reachable from only within the cluster, on a cluster-internal IP address (Figure 6),
- “NodePort” which exposes a service to the outside on a node’s IP and a static port,
- “LoadBalancer” which exposes a service externally using a cloud provider’s load balancer such as AWS ALB (Figure 5).

Multi-cluster connectivity will ensure direct connectivity for ClusterIP services across connected clusters without the need to expose a NodePort or LoadBalancer service.

7.1.3.2 Service Discovery/DNS

In a single cluster, an internal Kubernetes DNS server, usually “coredns”, provides each Pod and Service with a DNS record in the cluster domain. This allows for intra-cluster communication between Pods and Services using consistent DNS names instead of IP addresses. For example, “nginx.prod.svc.cluster.local”, where “nginx” is the name of the service, “prod” is the namespace, “svc” stands for the service resource, “cluster.local” is the domain of the cluster.

Service discovery is essential in a multi-cluster environment. Pods and Services in the interconnected clusters should be able to address one another by name. For this, the participating clusters would either extend the “cluster.local” domain or share a new global multi-cluster domain on which multi-cluster services can be registered. The created DNS records should be reachable from any participating cluster.

7.1.3.3 Network Policies

Network policies provide a native way to control traffic flow at the IP address or port level. They can enforce rules to allow traffic from specific Pods, Namespaces, or IP blocks within a cluster. It’s important to retain this ability when connecting multiple clusters. It is also important to draw a distinction between policy enforcement and policy propagation. Connectivity solution could enforce but might not automatically propagate Network Policy resources across clusters.

7.1.3.4 Encryption

By default, the traffic inside a Kubernetes cluster is not encrypted, however, certain CNIs offer encryption capabilities. With multiple clusters, all cross-cluster traffic should be encrypted using secure tunnel technology. This extra layer of security is essential as cross-cluster traffic flows over the Internet.

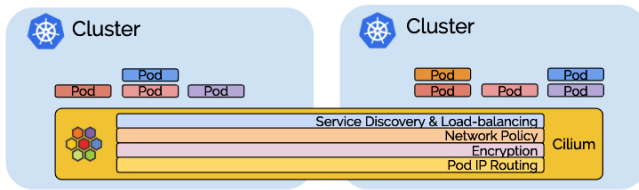


Figure 7. K8s networking concepts extrapolated to multi-clusters; adapted from [19]

To summarize, with multi-cluster connectivity, the goal is to fully preserve the native Kubernetes networking model and extend it beyond the boundaries of a single cluster. Figure 7 illustrates conceptually the extrapolation of K8s in-cluster networking to multiple clusters using the example

of Cilium.

7.1.4 Storage

In Kubernetes, Volumes are the base abstraction for storage architecture. Containers can request storage resources dynamically through volume claims. By default, a container's storage is temporary or non-persistent; it is removed as soon as the container consuming it is shut down.

PersistentVolumes (PV) is a K8s concept that allows data to "survive" even after containers shut down. Persistent Volumes are pieces of storage whose lifecycle is independent of Pods using them. PVs are created using options available at provider's site. For instance, AME clusters on AWS use EBS and EFS volumes, but AME clusters on Intermax use Ceph storage. This behavior is automated through StorageClasses and Provisioners.

Provisioning and migrating persistent storage in a multi-cluster environment is a separate challenge which is out of the scope of this research. However, some recommendations are given in 10.1.3.

7.2 Conceptual Multi-cluster Architectures

Mere inter-cluster connectivity does not solve the problem, there must be a good architecture to make the connectivity practical. Just as there are many ways to design an application architecture, there are also multiple ways to architect a multi-cluster Kubernetes environment depending on a use case. This sub-chapter looks at conceptual architectures with respect to Avisi use cases.

7.2.1 Multi-cluster segmentation architecture

As determined in Appendix A, chapter 1.5, jurisdiction compliance is one of the most critical use cases for multi-cluster on AME. This can be achieved through a segmentation architecture. As explained in

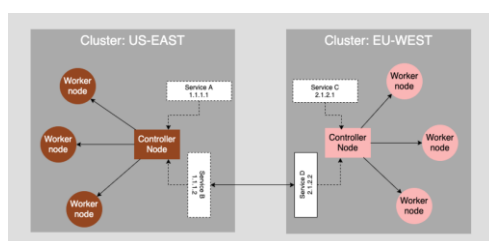


Figure 8. A multi-cluster segmentation architecture; from [26]

[26], in a segmentation architecture, an application gets separated into independent components, each typically represented as a Kubernetes service. Then, the services get allocated to interconnected clusters in different regions or clouds according to regulatory requirements and communicate with each other directly across clusters. The key benefit of such an architecture is that it allows application to enforce regulatory compliance across transnational

boundaries. AME can use this architecture to provide the platform to organizations that operate globally.

The figure below is an example of a segmentation architecture with two interconnected clusters. Service B in the US-EAST directly communicates with Service D in the EU-WEST.

7.2.2 Multi-cluster replication architecture

Replication is another approach to leverage K8s multi-cluster architecture. In a replication scenario, exact copies of a cluster are hosted in different regions and even clouds [26]. This architecture is also described in [27] as “multi-site active-active” where all clusters are read-write, and the data is synchronized in real time.

This type of design creates global HA in which case the Kubernetes environment can withstand a failure of an entire region. Further, this architecture can help achieve the lowest latency by physically locating workloads closer to end-users. This type of design is best suited for companies with critical workloads and data which is aligned with the industries Avisi is targeting. The main tradeoff of this architecture is high costs.

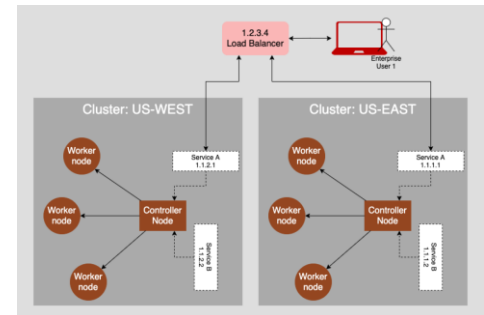


Figure 9. Service replication in a multi-cluster K8s architecture; from [26]

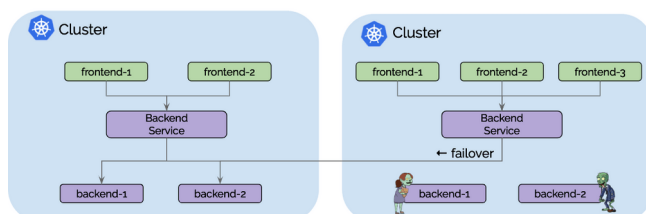


Figure 10. Multi-cluster service failover; from [18]

Figure 9 demonstrates two clusters in different clouds/regions that are virtually clones of each other. A cross-region Load Balancer is an entity responsible for routing requests to clusters based on their originating location. Even if the entire local cluster becomes unavailable, requests can still be satisfied by a remote one. Multi-cluster

networking can enhance this architecture and make it possible for individual services to failover to a remote cluster when they are not available locally as illustrated in Figure 10.

7.2.3 Multi-cluster active-passive architecture

Disaster recovery is the second most important use case for multi-cluster considered by Avisi. [27] describes a disaster recovery scenario using an active-passive multi-cloud K8s architecture. This architecture is identical to the previous one with the major difference being the second cluster is in the “standby” mode. Usually, it means that it is only used for periodic backups and occasional read-only operations. But as soon as the primary cluster fails, the standby one becomes fully active, and the service can be quickly switched with minimal interruptions. This is suitable for storage services with large data volumes.

7.2.4 Multi-cluster topologies: “Hub-and-Spoke” & “Full Mesh”

Previous architectures covered scenarios with two interconnected clusters. The use cases of Avisi might call for connecting three and more clusters. There are multiple ways to do it; one of them is hub and spoke topology (Figure 12). In this case, one cluster plays a role of a central hub that connects and routes traffic across other clusters connected to it. This is a cost-effective solution except the hub causes latency and is a single point of failure.

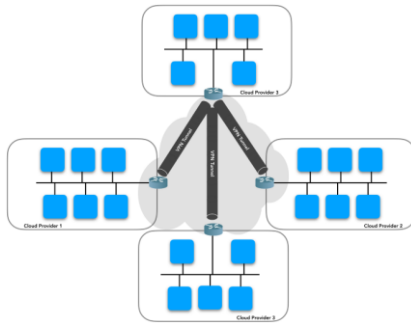


Figure 12. Multi-cluster hub-and-spoke topology; from [29]

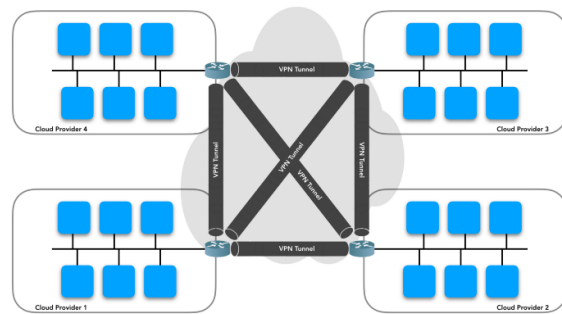


Figure 11. Multi-cluster full mesh topology; from [29]

In contrast, clusters could be interconnected into a full mesh (Figure 11) and instead of being conjoined through a central hub, each cluster is connected to every other participating cluster. Full mesh yields the greatest amount of redundancy but can be impractical due to being very expensive and extremely difficult to operate. Either hub-and-spoke or full mesh topology can be used in combination with segmentation or replication architecture depending on the case. The connectivity solutions proposed in this research support various centralized and decentralized topologies including the two described above.

7.2.5 Multi-cluster hierarchical federation

The architectures discussed previously mainly arrange clusters into a flat space as HA replicas, however, multi-cluster connectivity allows for much more advanced architectures. This becomes most powerful when dealing with a large number of clusters. An interesting multi-cluster architecture proposed in [28] describes a scenario where services are arranged in dozens of separate K8s clusters interconnected into a hierarchical structure illustrated below.

In the example shown in Figure 13, leaf clusters called deployment clusters are treated as HA replicas within a region; they are not aware of the other clusters and can only connect to resources inside

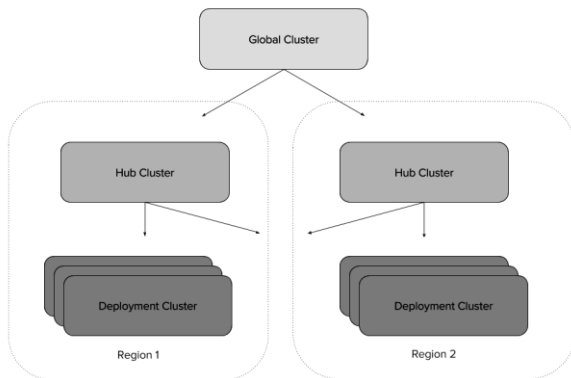


Figure 13. Multi-cluster hierarchical federation; from [30]

their own cluster. The second layer consists of hub clusters that can access all deployment clusters' services but are not aware of the other hub clusters. Hub clusters are designed to serve traffic to the leaf clusters in the same region, but they are also able to take traffic from hub clusters in different regions if required. At the top, there is a global cluster, ensuring a global view and enabling meta-monitoring. It can connect to all hub and deployment resources.

The main incentives for this topology stated in the source are the same as those of Avisi: service and data locality, fault isolation, HA. But on top of that, hierarchical architecture allows for other benefits such as partial network isolation and easier management.

7.3 Multi-cluster Orchestration

When the networking requirement is satisfied, the next logical step is seamless orchestration – distributing and managing workloads consistently and automatically across interconnected clusters from a central place. Just like in a single cluster, orchestration in a multi-cluster is a broad mechanism. It requires a form of synchronization between clusters' control planes to propagate K8s resources and achieve seamless multi-cluster scheduling and autoscaling. The current literature describes primarily three approaches to multi-cluster orchestration [12]:

1. Using a dedicated API server on a central cluster to coordinate configurations of multiple clusters from a set of extensions to traditional Kubernetes API (e.g., KubeFed, Mck8s)
2. GitOps approach relying on configurations stored in Git and using CI/CD pipelines to deploy to target clusters. (e.g., FluxCD)
3. Through a Virtual Kubelet approach in which a remote cluster can be mapped to a local cluster node. (e.g., Admiralty, Liko)

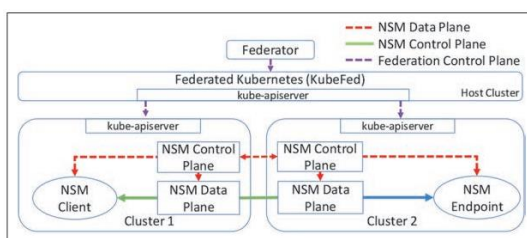


Figure 14. Combined multi-cluster solution based on NSM and KubeFed; from [31]

Ideally, a connectivity solution should be combined with a form of multi-cluster control plane to achieve a full-fledged multi-cluster environment. An example of such a combined approach is proposed in [29]. The described solution relies on NSM for cross-cluster connectivity and KubeFed for scheduling and orchestration (Figure 14).

Kubernetes Cluster Federation (KubeFed) is a K8s SIG Multi-cluster project, the official way to orchestrate a

multi-cluster environment. It aims to provide mechanisms to coordinate configuration of multiple clusters from a single set of APIs in a hosting cluster. Fundamentally, KubeFed offers building blocks for resource propagation – distribution of K8s resources to federated clusters. However, the main drawback of KubeFed and most other existing multi-cluster orchestration solutions is they not seamless – they force users to use new sets of API objects, for example, the FederatedDeployment (KubeFed) or a MultiClusterDeployment (Mck8s) instead of a native Deployment object. This subject is further addressed in 10.1.1.

In contrast, Lipo aims to work with existing Kubernetes ways. Instead of introducing a set of new APIs like KubeFed does, Lipo seamlessly extends cluster resources via a virtual-kubelet-based approach. The local cluster “sees” remote (connected) clusters as “big nodes”. A big node or a virtual node is equivalent to a physical node, hence it can be controlled by the native Kubernetes scheduler and controller manager (Figure 15).

Moreover, Lipo takes care of the entire multi-cluster topology lifecycle: not only on pod scheduling and offloading but also on cluster discovery and cross-cluster networking. Lipo is the only present solution that integrates connectivity and orchestration in one tool. For details, refer to Appendix B, chapter 2.4.

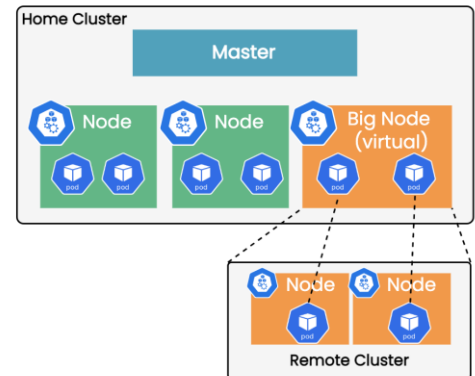


Figure 15. Lipo Big Node (Virtual Kubelet)

7.4 Conclusion

The core concepts of Kubernetes need to be extrapolated beyond a single cluster. The connectivity agent can help to extend network-related concepts such as routing, policies, and DNS from a single cluster to multiple clusters. Extending aspects like scheduling and orchestration should be the next logical step after connectivity, otherwise, deployments will have to be managed manually by cluster operators.

Multi-cluster connectivity opens the doors to building various architectures with K8s. Just as there are different application architectures, there are many ways to architect a multi-cluster K8s environment depending on a use case. This chapter covered a few conceptual architectures for the use cases of Avisi.

Prototyping was used as a research method (chapter 9), including for testing described architectures. Prototypes have proven that segmentation can be very efficient for jurisdiction compliance. Active-passive multi-cluster is an effective approach to deal with disasters. Global HA, low latency can be achieved through active-active replication. Finally, when dealing with many clusters, they can be arranged in a hierarchical structure for granular network isolation.

8 Technical Design

In this chapter, the condensed version of the Technical Design is discussed. The full version can be found in the Appendix D.

8.1 Introduction

Based on the results of the MCA described in 6.3.5, the final Technical Design will be represented by three architectures based on three different multi-cluster technologies: Linkerd, Liqo, and NSM – each being best-suited for different use cases. All three designs fully support multi-region and multi-cloud deployments, therefore AME clusters can be connected regardless of a region and cloud provider, granted that all prerequisites are met (e.g., LoadBalancer UDP support).

Due to limitations, this report does not cover a full spectrum of Technical Design's components but rather demonstrates its feasibility based on the example of two 'MUST' user stories: **US-5-F** (multi-cluster services) and **US-11-F** (encryption). These user stories showcase the core of the project – a secure network connectivity between different K8s clusters. The full evaluation of all user stories and design principles can be found in Appendix D.

8.2 Proposed Architecture 1 – L7 Multi-cluster connectivity with Linkerd

The current design is based on Linkerd version v2.11. Linkerd Multi-cluster is an extension to the Linkerd service mesh that enables transparent communications between services in different clusters through proxies, gateways, and service mirroring. Linkerd works at network Layer 7 and solves the first

case described in **DP-1**, enabling multi-cluster communication over HTTP, gRPC and other TCP-based protocols.

Figure 16 illustrates two AME clusters: cluster "Ireland" hosted on AWS in the eu-west-1 region, and cluster "Amsterdam" hosted on DigitalOcean in the ams2 datacenter. The Linkerd control plane is installed in both clusters using a shared trust anchor – a TLS certificate that will be used for cross-cluster traffic encryption.

Services on different clusters can reach each other directly by name. The **Linkerd control plane** is responsible for managing the service mesh as a whole. The **Linkerd data plane** comprises ultralight proxies that intercept all requests and provide security and observability features.

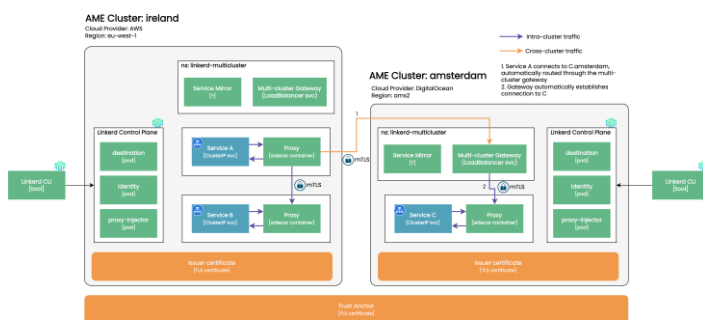


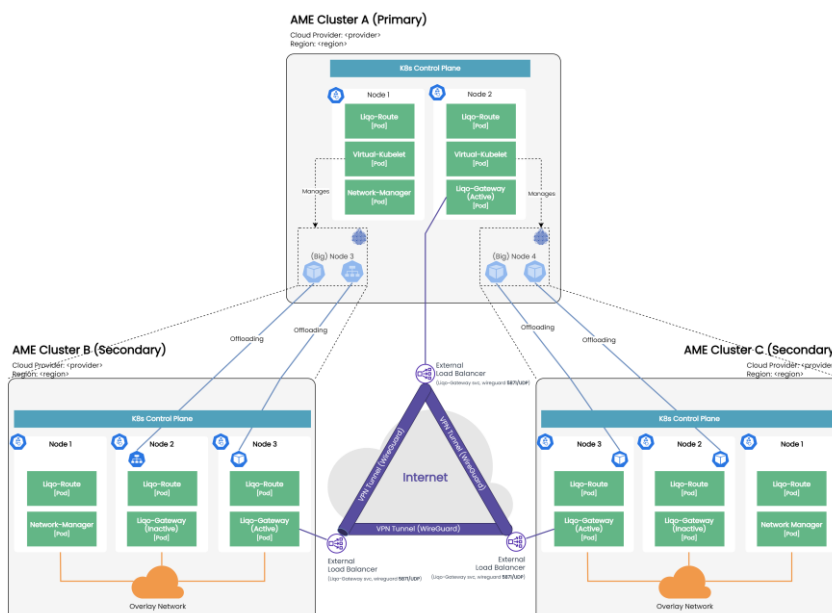
Figure 16. Linkerd Multi-cluster Architecture with two AME clusters

For US-5-F (connectivity), Linkerd guarantees connectivity between multi-cluster services with two main components: **Multi-cluster Gateway** and **Service Mirror**. The Multi-cluster gateways are exposed to the public Internet via a K8s service of type LoadBalancer; they provide target clusters a way to receive requests from source clusters and automatically route those requests to the correct Pods. Further, the Service Mirror component reflects (copies) the services between the clusters providing visibility into service names and enabling applications to address remote services directly.

For US-11-F (encryption), as shown in the diagram, all outgoing and incoming requests are intercepted by sidecar proxies before being routed to services. These proxies, among other things, guarantee traffic encryption. This is possible because Linkerd installations on the clusters share a trust anchor – a root TLS certificate that is used to generate the issuer certificates for each cluster in the topology. The proxies send certificate signing requests (CSR) to the Identity service that uses the issuer certificates to fulfil the requests. As a result of this process, all intra- and cross-cluster traffic is automatically authenticated and encrypted via mTLS.

8.3 Proposed Architecture 2 – L3 Multi-cluster connectivity with Ligo

The current design is based on Ligo version v0.3.2. Ligo allows to build a wide range of multi-cluster topologies: centralized and decentralized, with unidirectional or bidirectional resource sharing. This Technical Design showcases one possible architecture with Ligo called a “Supercluster” – a cluster of clusters. Ligo-based architecture solves the second case described in **DP-1**, providing workloads with a direct IP reachability at network Layer 3.



As demonstrated by Figure 17, the proposed architecture with Ligo provides multi-cluster connectivity and encryption through highly performant, secure and automatically managed VPN tunnels. Table 14 lists the services required to be reciprocally accessible by all Ligo clusters to establish multi-cluster connectivity through encrypted tunnels.

Figure 17. Ligo Multi-cluster Architecture (Ligo Supercluster) with three AME clusters

Service	Protocol	Default Port	Description
Liqo-Auth	HTTPS	443	Liqo service exposed through a LoadBalancer service or Ingress. It is used to authenticate incoming peering requests
Liqo-Gateway (WireGuard)	UDP	5871	Liqo service exposed through a LoadBalancer service. It is responsible for VPN tunnels between clusters
Kubernetes API Server	HTTPS	443	A standard Kubernetes API server used in combination with etcd (storage) for the creation and replication of required resources. On AME, exposed through an Ingress

Table 14. Services used by Liqo clusters to establish connectivity

For US-5-F (connectivity), this flattens the L3 network between K8s clusters allowing Pods and Services on different clusters to communicate directly to each other. On top of that, Liqo automates the deployment of multi-cluster services (service replication) by providing seamless dynamic scheduling and spreading service Endpoints across multiple clusters through the process called Offloading. In this process, the remote clusters are represented as “Big” nodes in the local cluster; they are controlled by a Virtual Kubelet and act like regular K8s nodes. In this topology, the peering is unidirectional, meaning that the resources are shared only from secondary clusters to a primary cluster and not vice versa. Depending on requirements, bidirectional peering can be established making it possible to share resources both ways.

For US-11-F (encryption), all cross-cluster traffic flows through encrypted VPN tunnels using the WireGuard technology. WireGuard is an incredibly performant tunnel technology that uses state-of-the-art cryptographic protocols and primitives such as ChaCha20, Poly1305, Curve25519, etc. [30]. Liqo-Gateway automatically manages the entire lifecycle of the tunnels; additional configuration can be done to achieve the best performance such as installing WireGuard kernel module on AME cluster nodes.

8.4 Proposed Architecture 3 – L2/L3 Point-to-Point Multi-cluster connectivity with NSM

The proposed design is based on NSM version v1.2.0. From research findings, using NSM for multi-cluster connectivity is significantly more complex than Linkerd or Liqo, thus, it is not recommended to employ it for a generic use case. However, this architecture addresses the third case described in DP-1 and supports lower-level networking features (L2/L3). In addition, as it was described in 6.3.5, NSM uniquely solves the use cases of individual (point-to-point) and cross-runtime-domain connections, therefore, AME should keep the project in sight. Upcoming versions of NSM will include enhancements that will improve its integrability with AME clusters and bring the complexity down.

- Out-of-the-box implementation of a virtual Layer 3 (vL3) Network Service which will provide a common routing for workloads running in different clusters or cloud providers.
- Integration of NSM and Calico-VPP which will allow the two to share the same VPP forwarder instance on a node improving overall performance.
- Advanced healing features that will act more robustly when the NSM control plane or NS data path becomes unavailable.

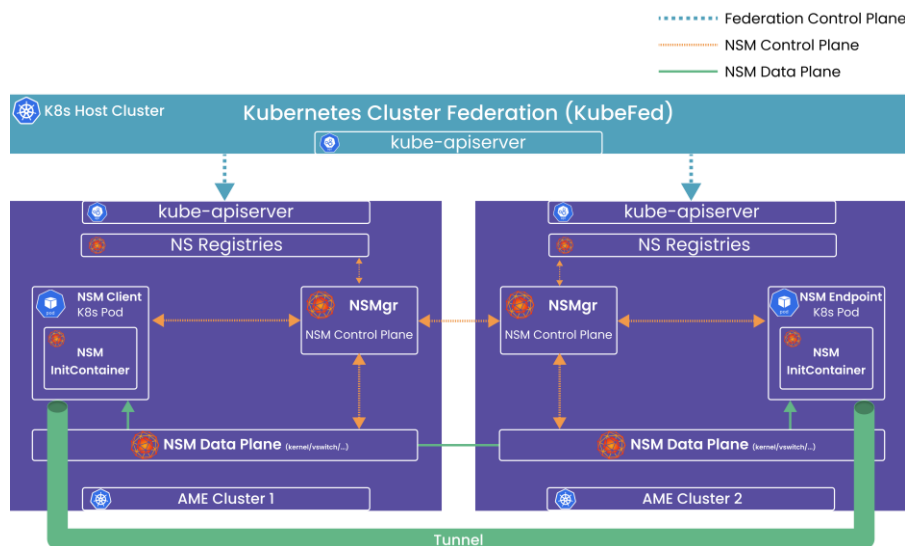


Figure 18. Multi-cluster architecture with NSM and KubeFed

Figure 18 illustrates a high-level architecture with two AME clusters and two individual Pods interconnected via NSM.

The architecture comprises four primary components:

- the **NSM control plane** (NSMgr (Network Service Managers)) which is responsible primarily for connection

management (i.e., injection and removal of network interfaces),

- the **NSM data plane** which is, essentially, a forwarder (router) component,
- the **NS Registries** that use etcd to store information about forwarders, Network Services and Endpoints,
- the connected Pods (**NS Client** and **NS Endpoint**).

KubeFed is not a strict requirement; its purpose in this architecture is solely to simplify deployments to multiple clusters. Therefore, it can be either replaced with another multi-cluster control plane solution such as Mck8s, Admiralty or completely excluded with no effect on NSM network connectivity. The integration of NSM and KubeFed will not be tested during this project.

For US-5-F (connectivity), NSM can provide a distributed vL3 domain (as a Network Service) that allows individual workloads (Pods) in different clusters and cloud providers to communicate via IP. The mechanism through which NSM achieves this is completely different from Linkerd and Liqo. Instead of utilizing the existing internal K8s networking (CNI), NSM introduces a separate connectivity domain completely independent from a CNI running inside a cluster. The connectivity between two Pods can be requested and released at runtime with annotations. NSM injects new network interfaces into the Pods on both sides whenever connectivity is requested, through which IP packets are forwarded.

For US-11-F (encryption), NSM's model of a Network Service aims to provide not only connectivity but also security and observability features at L3 and above for individual workloads. All these features can be composed into Network Services, including VPN termination to enable traffic encryption. NSM provides full flexibility in choosing a data plane forwarder and VPN mechanisms because all components are pluggable. At the same time, the underlying implementation is more complex as it involves low-level networking concepts. An example of possible implementation is described in [31] as "Kernel to WireGuard to Kernel connection". In future releases, AME would be able to use Calico-VPP with NSM.

8.5 Conclusion

The MCA showed that different connectivity solutions work at different layers of network. According to DP-1, a connectivity layer should be chosen based on applications' needs. In this Technical Design, three different architectures have been proposed.

The first architecture is based on the Linkerd service mesh's multi-cluster extension. It provides Layer 7 connectivity for services that communicate over TCP-based protocols such as HTTP, gRPC, etc. Linkerd achieves this via proxies, service mirrors, and publicly exposed multi-cluster gateways. Linkerd-based design conforms to all design principles formulated in 6.4 and all AME criteria formulated in the MCA. This architecture is simple and universal.

The second architecture is based on Layer 3 connectivity via VPN tunnels managed by Lipo. It enables direct Pod-to-Pod and Pod-to-Service IP reachability across clusters. The design based on Lipo conforms all design principles and most AME criteria. This architecture is more sophisticated than the one with Linkerd as it enables multi-cluster scheduling besides networking.

The third architecture is powered by Network Service Mesh. Its purpose is to provide per-workload connectivity for applications that require lower-level network features or support for non-standard protocols that use Ethernet/IP payloads. NSM enables granular L2/L3 connectivity by injecting secondary network interfaces into Pods on each side. The NSM-based design conforms most design principles and satisfies most AME criteria.

All three architectures provide multi-cluster connectivity (at different layers) necessary to handle jurisdiction compliance, DR, HA, and more.

9 Proof of Concept

Prototyping was used as one of the research methods in this project. Since not all capabilities could be proven through documentation, smaller prototypes were built to evaluate individual MCA criteria for the shortlisted solutions: Submariner, Liko, Linkerd, NSM. Furthermore, three PoC prototypes were built to answer the research question 8 and validate the final architectures proposed in chapter 8. This chapter summarizes the challenges during prototyping and the results they yielded. For a complete prototype log, refer to Appendix E.

9.1 Prototyping Environment

Two types of prototyping environments were used – local and staging. The local environment consisted of test Kubernetes clusters created locally on my laptop using tools such as minikube, Kind and k3d. This environment was used for day-to-day tests on individual features.

The staging environment is a real AME environment which almost completely reflects the production configuration. This environment was used for the final PoC prototypes and to validate a multi-cloud and multi-region setup. For this, clusters were created in with two different providers and regions: AWS (eu-west-1) and DigitalOcean (ams3).

To prove multi-cluster connectivity and test multi-cluster (East-West) services, mock applications were deployed using scenarios such as replication and segmentation described in 7.2.

9.2 Linkerd

The final prototype with Linkerd was built using two AME staging clusters. it is consistent with the architecture described in 8.2.

All Linkerd components were deployed using Linkerd CLI; no issues have been discovered during this process. A sample microservices application (provided by [32]) was used to validate the following features: multi-cluster connectivity, encryption, authorization policy, traffic split and distributed tracing.

In addition, the PoC with Linkerd successfully demonstrated automated multi-cluster per-service failover, multi-cluster communication for StatefulSets as well as proxying traffic for Postgres. The prototype with Linkerd has proven its simplicity and robustness since almost no additional configuration was required on AME clusters compared to Liko and NSM.

9.3 Liko

The PoC with Liko fully reflected the architecture proposed in 8.3. It was built using two AME clusters on AWS and one cluster in DigitalOcean (DO). Several challenges have been discovered when prototyping with the current version of Liko (v0.3.2). Challenges included:

- Inability to use the standard (recommended by Liko) deployment process with AME. To solve it, an alternative installation method tailored to AME was tested and described in Appendix D, chapter 4.2.2.
- Inability to patch Calico on AME to ignore Liko-managed network interfaces using the instructions provided by Liko. An alternative approach is further addressed in the above appendix.
- DigitalOcean not supporting a UDP Load Balancer required by Liko-Gateway. This is a provider constraint; Liko-Gateway cannot be exposed with a DO LB as of now.

Overall, multi-cluster connectivity using Liko-based architecture was successfully tested in the following scenarios: a. PostgreSQL replication across three clusters (1 replica per cluster), b. sample cloud-native application with 10 microservices (provided by [33]) segmented across 3 clusters using selective offloading strategy.

9.4 NSM

At the time of writing this report, the PoC with NSM connectivity was tested using only local clusters created with KinD and only with “ping” between the NSC (alpine) and NSE (ICMP responder) Pods. The connections were successfully tested using the following basic examples defined in the “deployments-k8s” GitHub repository that belongs to Network Service Mesh: a. Kernel to VXLAN to Kernel; b. Kernel to WireGuard to Kernel.

The main challenge faced during prototyping with NSM is the absence of an out-of-the-box implementation of a virtual Layer 3 (vL3) Network Service that can be used to provide a common routing domain to which workloads in different clusters or cloud providers may attach. The vL3 implementation will be released in the upcoming version of NSM (v1.3.0) in April 2022 which would allow to test multi-cluster connectivity using real clusters in the staging environment of AME and with more sophisticated scenarios such as Postgres database replication.

Furthermore, the PoC did not include the integration of NSM with KubeFed described in the design because the primary focus was on testing network connectivity.

10 Advice on Multi-cluster Challenges

Along with numerous benefits, multi-cluster architectures bring new operational challenges. This chapter focuses on answering the research question 9 – “What are the challenges associated with complexity, security, and observability in a Kubernetes multi-cluster environment, and what could be done to overcome these challenges?”. The recommendations are based on the most recent publications from CNCF, Kubernetes SIG Multicluster and Gartner.

10.1 Complexity

Added complexity is one of the fundamental challenges that go with multi-cluster architectures. Multi-cluster creates an array of questions related to deployments, storage, load balancing, debugging and more.

10.1.1 Multi-cluster Deployments

Multi-cluster networking solutions, except for Liko, do not deal with deploying workloads across multiple clusters. A form of multi-cluster control plane is required to facilitate multi-cluster rollout strategies, orchestration, and workload distribution. There are primarily two ways AME can address this challenge – through extending the existing internal GitOps practices to support multi-cluster setups or through dedicated multi-cluster control plane tools.

The first method is preferred for at least two reasons. First, AME already embraces GitOps approach for individual clusters by using FluxCD to install applications inside clusters. Second, this approach does not introduce any extra APIs unlike the ones described further. If configured properly, GitOps can represent an elementary multi-cluster control plane. However, this approach is minimalistic and will not provide sophisticated functionality like dynamic Pods placement, cross-cluster migrations, DR etc. For this, a more advanced multi-cluster control plane is required.

The table below lists several open-source projects that can be used by AME for multi-cluster orchestration in combination with connectivity provided by Linkerd, Liko or NSM.

Project	Description
Admiralty	An open-source project that provides a system of Kubernetes controllers that intelligently schedule workloads across clusters. [34]
Shipper	An open-source extension for Kubernetes that enable sophisticated rollout strategies and multi-cluster orchestration. It enables multi-cluster, multi-region, and multi-cloud deployments and release management. [35]
KubeFed	Kubernetes Cluster Federation. An official K8s project that allows to coordinate the configuration of multiple clusters from a single set of APIs in a hosting cluster. [36] Makes multiple separate clusters appear as a single cluster from an operational standpoint. [8]
mck8s	mck8s, short for multi-cluster Kubernetes, allows you to automate the deployment of multi-cluster applications on multiple Kubernetes clusters by offering enhanced

configuration possibilities and scheduling policies (e.g., resource-based, network-based, etc.) [37]

Table 15. Multi-cluster control plane solutions

10.1.2 Multicluster Load-Balancing

When services are deployed across multiple clusters, the next logical step is to serve and distribute requests across those clusters whether they are on-prem or in cloud. A Global Server Load Balancing (GSLB) solution is required to direct HTTP requests across K8s local load balancers (e.g., Ingress controller instances) based on the health of targets (Pods). In many cases, when clusters are in different regions, a GSLB should be locality-aware – direct requests based on their originating location to minimize latency and provide the best user experience.

It is challenging to satisfy these criteria due to the absence of standardized methods to provision and manage GSLBs in a Kubernetes-native way. With traditional servers, LSB solutions have mostly been a task for proprietary infrastructure providers that configured and maintained those tools by siloed network teams. This is not a cloud-native friendly way. And for AME, as a provider-agnostic platform, it is of special importance to avoid using proprietary solutions.

To address this challenge, AME could integrate support for K8GB – a cloud-native Kubernetes Global Balancer. As explained in [38], K8GB is a CNCF-sandbox project with a focus on providing an open source, cloud-native global load balancing for Kubernetes. It allows to deploy a GSLB with a single Kubernetes custom resource (Gslb kind), performs balancing on timeproof DNS protocol, does not induce any dedicated management cluster and no single point of failure, uses K8s-native health checks for load balancing decisions.

Global locality-aware load balancing has its own risks and should be implemented with precautions. When this mechanism is used to failover to a remote cluster if the local one is unable to serve requests, there are at least two risks:

- As the cluster is remote, latencies are going to spike. If services have configured timeouts or SLOs set, it is likely those are going to be breached and produce cascading failures.
- Depending on how overhead is configured, a remote cluster jumping in 2x load all of the sudden might crash that cluster as well.

To mitigate these risks, it is recommended to perform extensive manual testing before automating this mechanism.

10.1.3 Storage

The AME platform hosts many stateful workloads – applications that save data to persistent disk storage that does not get destroyed when a Pod consuming it is terminated. Databases and key-value stores are examples of such stateful workloads. According to [8], deploying and managing stateful applications in a Kubernetes multi-cluster environment adds another layer of complexity related to storage and data dependencies.

In Kubernetes, the binding between a Pod and persistent storage is managed by persistent volume claims (PVC). When a Pod running a stateful application migrates from one cluster to another, it is challenging to reestablish the correct binding between a migrated Pod and its data. Currently, there are no standardized processes within K8s as it does not define any federated method for specifying PVCs. There are several considerations that could be useful for AME.

First, the architecture with Liqo proposed in 8.3 can simplify deploying and migrating stateful applications across multiple clusters. In the upcoming versions, Liqo will introduce support for a virtual storage class that maps a virtual PV to an actual PV available through a cluster's storage classes. This functionality will allow support for stateful workloads on Liqo-enabled clusters. To support the migration of a stateful Pod across clusters, Liqo will implement an ability to move PVs between clusters. The first expected implementation will be done through Restic – a backup software that will allow to create a snapshot in a local cluster and restore it in a remote one [39]. This concept is unique to Liqo; other solutions considered in this research do not offer this capability as of now.

Another consideration for the storage challenge is using a container-native storage (CNS). This is a new generation of storage solutions designed specifically for containerized workloads, some of which support Kubernetes multi-cluster architectures. For instance, Portworx offers a distributed persistent storage platform for multi-cluster stateful applications. In addition, it facilitates mechanisms for multi-/hybrid-cloud migrations and disaster recovery [8].

10.2 Security

DP-2 already emphasized the importance of adopting a zero-trust security model in a multi-cluster environment. The motivation behind it is simple – more clusters mean a larger attack surface area. Adopting a zero-trust security model is essential to protecting such an environment.

Each cluster has its own set of security-related resources such as Secrets, Certificates, Roles, etc. AME uses cert-manager to provision and manage TLS certificates automatically. As of now, cert-manager is only able to create and manage these resources in a single cluster. One possible solution is to federate these resources through KubeFed – multi-cluster control plane solution that supports federated CRDs. Federation would allow to propagate Certificates, Secrets, and other security resources across clusters. This option remains unexplored; as of now, there is not enough data to evaluate how well it would work.

Another consideration for distributing Secrets and Certificates is an external dedicated secret management solution that can be accessed by multiple Kubernetes clusters. There are projects like external-secrets [40] that allow to read information from third-party secret management solutions such as AWS Secret Manager or Hashicorp Vault and automatically inject the values as Kubernetes Secrets.

10.3 Observability

As explained in [41], observability refers to the ability to gain insights into what is happening inside a system based on the external data exposed by that system. AME already enables rich observability for individual clusters through log and metrics collection and visualization using Prometheus, Grafana, Loki. The challenge of observability in a multi-cluster environment revolves around a so-called “single pane of glass” – a centralized entity that can collect and display monitoring data across all clusters. To achieve this, AME can use open-source solutions like Cortex or Thanos to aggregate data from multiple Prometheus deployments.

In addition, the components that provide multi-cluster connectivity themselves need to be monitored continuously to be able to react to connectivity losses and other incidents quickly. To address this, AME should configure Prometheus instances to scrape metrics exposed by those components. As of now, among proposed solutions, only Linkerd exposes out-of-the-box Prometheus metrics. It is expected that metrics will also be available for Liqo and NSM in the upcoming versions.

11 Conclusions and Reflection

Kubernetes multi-cluster architectures are an important next step in the evolution of the AME platform. This research aimed at investigating how these emerging architectures can provide the customers of Avisi with flexibility and choice over the geographical locations and cloud providers hosting applications and data. Given the plans of AME to continue expanding support for more cloud providers and regions, Kubernetes multi-cluster architectures will open up a fundamentally new class of opportunities for the platform's users. These architectures drastically simplify connecting, deploying, and migrating applications across different geographic regions and cloud providers. This gives versatility in solving such problems as jurisdiction compliance, disaster recovery, high availability, multi-cloud and more.

Multi-cluster architectures involve primarily two challenges: network connectivity and orchestration across clusters. This research focused mostly on network connectivity. Nine multi-cluster connectivity solutions have been considered. The research has shown that although the solutions do overlap in many ways, they focus on different use cases and utilize different underlying technology. First, a constraints analysis was conducted which focused on integrability with the current platform. At this stage, 5 out of 9 solutions have been eliminated. Further, a comprehensive multi-criteria analysis was performed to evaluate the remaining solutions against 20 criteria of AME. The criteria looked at aspects such as multi-cluster service discovery, routing, traffic control, and connection security. The MCA revealed that no single solution can cover all use cases, and the choice depends on application needs. As a result, the final Technical Design proposes three different architectures that provide multi-cluster connectivity at different layers.

The first design discussed in 8.2 is based on the multi-cluster extension to the Linkerd service mesh. It is the most mature (CNCF-graduated) and simplest solution to establish multi-cluster connectivity at Layer 7. It is best suited for applications that communicate over TCP-based protocols such as HTTP, gRPC, WebSocket, etc. Followed by a design based on Liko explained in 8.3. This design extends Kubernetes Layer 3 networking to multiple clusters allowing Pods and Services in those clusters to communicate directly through secure VPN tunnels. Additionally, Liko provides seamless workload scheduling across clusters. Finally, the design proposed in 8.4 would allow providing individual workloads with L2/L3 connectivity across clusters. It is best suited for applications that require support for lower-level networking features or non-standard protocols that use Ethernet/IP payloads. Furthermore, all proposed designs follow the design principles defined and agreed with the Platform Architect.

Throughout the project, multiple prototypes have been built locally and in the AME staging environment which almost completely reflects the production configuration. Not only did the prototypes serve as a proof-of-concept proving the designs in practice, but they also helped to identify more technical challenges related to multi-cluster deployments, load balancing, and storage. As a result, additional tailored advice was formulated for the future.

To conclude, Kubernetes multi-cluster architectures are novel and complex. The designs and recommendations delivered in this research will minimize the complexity of implementing and operating these architectures on AME. Consequently, the customers of AME will gain exceptional versatility to deploy their applications and data across different regions and cloud providers.

11.1 Product Reflection

Overall, the prototypes successfully validated most acceptance criteria. The delivered designs follow the design principles established during research and agreed with the Platform Architect thus providing guarantees of integrability with the current platform. However, due to time constraints, some work could not be completed. This chapter talks about important steps that must be considered by AME to proceed with the implementation based on the proposed designs.

Prototyping with Linkerd has proven Linkerd's maturity, simplicity, and robustness. There were no significant issues when deploying Linkerd components and linking clusters; the service reflection process was straightforward, and connectivity was stable. In addition, I was able to test the most recent feature that has just been released by Linkerd in March 2022 – automated multi-cluster per service failover. I successfully tested this feature with two AME clusters in AWS. The only recommendation is to further investigate how Linkerd Service Mirror changes the names of reflected services and how it might impact applications that use internal K8s DNS (see Appendix E, chapter 5.4.3.).

Although the developers of Liko claim it is mature enough for production use, it is recommended that AME performs additional extensive tests. At the time of writing this report, the Liko architecture was tested successfully only with single-node AME clusters in AWS. The traffic is correctly sent and received across clusters by Liko gateways on each side, however, the inter-node traffic does not proceed from the node running the liko-gateway to other nodes in the cluster. After several debugging sessions and discussions with the Liko team, it is presumed that the packets destined for liko.vxlan interfaces get dropped possibly due to Calico configuration or the underlying network/security groups configurations in AWS. Further investigation is required to address the issue and be able to use Liko on multi-node clusters. Furthermore, due to the same issue, the Liko prototype did not test the Liko-gateway failover process which is critical for a resilient topology. When the multi-node problem is addressed, AME should test Liko with multiple gateway instances. The test should include unexpected termination of a node running the active Liko-gateway instance in which case the tunnel must be automatically reconciled by the standby replica of the Liko-gateway.

Additionally, basic tests have been performed to evaluate how Liko Resource Offloading can work in conjunction with the deployment tool currently used by AME – FluxCD. The tests were successful and showed that Liko and FluxCD can be a powerful combination to deploy applications across clusters. However, no real use cases were used; AME should follow through and investigate Liko Selective Offloading which in combination with FluxCD could enable fully automated multi-cluster deployments to handle a case like jurisdiction by configuring cluster affinities based on the cluster's

region/provider. Furthermore, there was not enough time to sufficiently evaluate another important feature of Liqo as it has just been released in v0.4 (end-March 2022). The Liqo multi-cluster PVC and volume migration capabilities hold a lot of potential to simplify the deployment and migration of stateful workloads across clusters. Since AME clusters host a lot of stateful workloads, the feature could be highly beneficial, therefore, AME should test it further.

The NSM-based multi-cluster connectivity was only tested using local clusters created with KinD and only using a basic test scenario that included an alpine image client (NSC) and an ICMP-responder endpoint (NSE). The current NSM configuration could be complicated due to the absence of OOTB implementation of a virtual Layer 3 Network Service. This implementation will be added in the next NSM release (v1.3.0) which is scheduled for April 4, 2022. When this happens, AME would be able to conduct more sophisticated tests. NSM roadmap also includes integration with Calico-VPP which would allow NSM and Calico to share a single forwarder and improve overall performance. AME should consider this integration when using NSM in the future. Furthermore, the proposed integration with NSM as a multi-cluster connectivity agent and KubeFed as a multi-cluster orchestrator has not been explored in the prototype.

Another important test that could not be finished at the time of writing this report is a multi-region and multi-cloud setup. A prototype that connects AME clusters in AWS with a cluster in DigitalOcean could not be fully completed. It is very important to test all three solutions in real multi-region and multi-cloud configurations, ideally, using real applications. This test should also include gathering information about the network (latency, throughput) and performance (CPU, memory) overhead for the user story US-17-NF which could not be evaluated as of now.

Further, although this research made sure the proposed solutions integrate well with the configuration of AME infrastructure, it did not dive further into which component of AME backend architecture should be responsible for “activating” multi-cluster connectivity. Options to consider are cluster-controller or addon-controller. Liqo requires certain adjustments to the core infrastructure (e.g., Calico CNI), therefore, it is important to find a proper place. Investigating how multi-cluster connectivity should be activated from the end-user perspective was also excluded from the scope (US-16-NF) and is a part of future work for AME.

Finally, by the end of the project, internal discussions suggested that the proposed multi-cluster connectivity solutions can be used not only as a customer-facing feature but also internally to address some problems within the root architecture of AME. This was not investigated further due to time constraints. AME should consider which of the connectivity solutions is best suited for the root cluster needs and build a PoC to evaluate the feasibility and stability of such integration.

11.2 Self-Reflection

This graduation project was incredibly complex from a technical standpoint. Virtually the entire technical part of the project is related to cutting-edge technology and concepts that I grasped on the fly. Thanks to this I was able to gain invaluable knowledge. A complete self-reflection on the project can be found in Appendix G.

List of Figures and Tables

Figures

Figure 1. AME Kubernetes-in-Kubernetes architecture overview	16
Figure 2. Analyzing the AME problem using 5 Why's	20
Figure 3. K8s Multi-cloud and multi-region distribution; adapted from [49]	29
Figure 4. Multi-cluster with separate masters and workers; adapted from [25]	37
Figure 5. Kubernetes LoadBalancer Service; from [28]	38
Figure 6. Kubernetes ClusterIP Service; from [27]	39
Figure 7. K8s networking concepts extrapolated to multi-clusters; adapted from [19]	40
Figure 8. A multi-cluster segmentation architecture; from [26]	40
Figure 9. Service replication in a multi-cluster K8s architecture; from [26]	41
Figure 10. Multi-cluster service failover; from [18]	41
Figure 11. Multi-cluster full mesh topology; from [29]	42
Figure 12. Multi-cluster hub-and-spoke topology; from [29]	42
Figure 13. Multi-cluster hierarchical federation; from [30]	43
Figure 14. Combined multi-cluster solution based on NSM and KubeFed; from [31]	43
Figure 15. Liqo Big Node (Virtual Kubelet)	44
Figure 16. Linkerd Multi-cluster Architecture with two AME clusters	45
Figure 17. Liqo Multi-cluster Architecture (Liqo Supercluster) with three AME clusters	46
Figure 18. Multi-cluster architecture with NSM and KubeFed	48

Tables

Table 1. Sub-questions, methods, and chapter references	10
Table 2. Design Constraints for AME multi-cluster solution	18
Table 3. Shortlist of stakeholders for requirements gathering	23
Table 4. Business-related User Stories	24
Table 5. Functional User Stories	25
Table 6. Non-functional User Stories	26
Table 7. Research Spikes	28
Table 8. Overview of active multi-cluster connectivity projects	31
Table 9. Constraints Evaluation of Multi-cluster Solutions	31
Table 10. MCA scores explanation	33
Table 11. MCA criteria weight	33
Table 12. Multi-cluster solutions MCA – total scores	33
Table 13. Design principles for a K8s multi-cluster architecture on AME	36
Table 14. Services used by Liqo clusters to establish connectivity	47
Table 15. Multi-cluster control plane solutions	53

List of Abbreviations

Abbreviation	Explanation
AME	Avisi Managed Environments
API	Application Programming Interface
AWS	Amazon Web Services
AZ	Availability Zone
BYOC	Bring Your Own Cloud
CIDR	Classless Inter-Domain Routing
CLI	Command-line Interface
CNCF	Cloud-Native Computing Foundation
CNI	Container Network Interface
CNS	Container Native Storage
CPU	Central Processing Unit
CRD	Custom Resource Definition (Kubernetes)
CSI	Container Storage Interface
CSR	Certificate Signing Request
DNS	Domain Name System
DO	DigitalOcean
DR	Disaster Recovery
EKS	Elastic Kubernetes Service
gRPC	Google Remote Procedure Call
GSLB	Global Server Load Balancing
HA	High Availability
IaC	Infrastructure-as-Code
ICMP	Internet Control Message Protocol
IPS	Intrusion Prevention System
IPSec	Internet Protocol Security
K8s	Kubernetes
L3, L7	Layer3, Layer 7 (Networking)
LAN	Local Area Network
LB	Load Balancer
mTLS	Mutual Transport Layer Security
NAT	Network Address Translation
NS	Network Service
NSM	Network Service Mesh
OOTB	Out of the Box
OSI	Open System Interconnection
P2P	Peer-to-Peer (Networking)
PaaS	Platform-as-a-Service
PoC	Proof of Concept
PV	Persistent Volume
PVC	Persistent Volume Claim
RAM	Random-Access Memory
RBAC	Role-Based Access Control
RUP	Rational Unified Process
SIG	Special Interest Group
SSH	Secure Shell
TCP	Transmission Control Protocol
TD	Technical Design
TLS	Transport Layer Security
UI	User Interface
VM	Virtual Machine
VPN	Virtual Private Network

Reference List

- [1] Flexera, "2021 State of the Cloud Report," 2021. [Online]. Available: <https://info.flexera.com/CM-REPORT-State-of-the-Cloud>. [Accessed 10 12 2021].
- [2] L. Larsson, "Cloud-agnostic third party managed Kubernetes services – the unexploited opportunity," 3 June 2021. [Online]. Available: <https://elastisys.com/cloud-agnostic-third-party-managed-kubernetes-services/>. [Accessed 3 December 2021].
- [3] Datadog, "8 Surprising Facts About Real Docker Adoption," June 2018. [Online]. Available: <https://www.datadoghq.com/docker-adoption>. [Accessed 5 December 2021].
- [4] Red Hat, "What is container orchestration?," 2 December 2019. [Online]. Available: <https://www.redhat.com/en/topics/containers/what-is-container-orchestration>. [Accessed 5 December 2021].
- [5] J. Withers, "What Is Kubernetes? An Introduction to the Wildly Popular Container Orchestration Platform," 26 May 2021. [Online]. Available: <https://newrelic.com/blog/how-to-relic/what-is-kubernetes>. [Accessed 5 December 2021].
- [6] H. Virdó, "What Is Immutable Infrastructure?," 26 September 2017. [Online]. Available: <https://www.digitalocean.com/community/tutorials/what-is-immutable-infrastructure>. [Accessed 5 December 2021].
- [7] Red Hat, "What is GitOps?," 3 May 2021. [Online]. Available: <https://www.redhat.com/en/topics/devops/what-is-gitops>. [Accessed 10 January 2021].
- [8] Gartner, Inc., "Assessing Patterns for Deploying Distributed Kubernetes Clusters," 8 April 2020. [Online]. Available: <https://www.gartner.com/en/documents/3983221/assessing-patterns-for-deploying-distributed-kubernetes->. [Accessed 12 January 2022].
- [9] CloudZero, "Cloud Agnostic: What Does It Really Mean And Why Do You Need It?," 13 August 2021. [Online]. Available: <https://www.cloudzero.com/blog/cloud-agnostic>. [Accessed 25 January 2022].
- [10] C. Tozzi, "Why Cloud Kubernetes Is Not as Vendor-Agnostic," 26 November 2020. [Online]. Available: <https://www.f5.com/company/blog/why-cloud-kubernetes-is-not-as-vendor-agnostic-as-it-seems-and-what-to-do-about-it>. [Accessed 25 January 2022].

- [11] A. Robinson, "Gotchas & Solutions Running a Distributed System Across Kubernetes Clusters," 20 December 2018. [Online]. Available: <https://www.cockroachlabs.com/blog/experience-report-running-across-multiple-kubernetes-clusters/>. [Accessed 27 January 2022].
- [12] G. Arbezano, "Simplifying multi-clusters in Kubernetes," 12 April 2021. [Online]. Available: <https://www.cncf.io/blog/2021/04/12/simplifying-multi-clusters-in-kubernetes/>. [Accessed 15 December 2021].
- [13] Kubernetes, "Cluster Networking," [Online]. Available: <https://kubernetes.io/docs/concepts/cluster-administration/networking/>. [Accessed 27 January 2022].
- [14] Red Hat, "A brief overview of the Container Network Interface (CNI) in Kubernetes," 29 April 2021. [Online]. Available: <https://www.redhat.com/sysadmin/cni-kubernetes>. [Accessed 27 January 2022].
- [15] Submariner, "Submariner," 2021. [Online]. Available: <https://submariner.io/>. [Accessed 24 January 2022].
- [16] Skupper, "Skupper," [Online]. Available: <https://skupper.io/>. [Accessed 26 January 2022].
- [17] Ligo, "Ligo," [Online]. Available: <https://ligo.io/>. [Accessed 26 January 2022].
- [18] Cilium, "Deep Dive into Cilium Multi-cluster," 18 March 2019. [Online]. Available: <https://cilium.io/blog/2019/03/12/clustermesh>. [Accessed 26 January 2022].
- [19] Istio, "Multicluster Deployments," [Online]. Available: <https://istio.io/v1.2/docs/concepts/multicluster-deployments/>. [Accessed 26 January 2022].
- [20] Linkerd, "Multi-cluster communication," [Online]. Available: <https://linkerd.io/2.11/features/multicluster/>. [Accessed 26 January 2022].
- [21] The Network Service Mesh, "Network Service Mesh," [Online]. Available: <https://networkservicemesh.io/>. [Accessed 2022 January 26].
- [22] kEdge Authors, "kEdge," [Online]. Available: <https://github.com/improbable-eng/kedge>. [Accessed 26 January 2022].
- [23] Consul Authors, "Multi-Cluster Federation Overview," [Online]. Available: <https://www.consul.io/docs/k8s/installation/multi-cluster>. [Accessed 26 January 2022].
- [24] Ligo Team, "CNCF Turin," in *Unleashing the multi-cluster potential with Ligo*, Turin, 2021.
- [25] Kubernetes, "Kubernetes Documentation," [Online]. Available: <https://kubernetes.io/docs/home/>. [Accessed 6 February 2022].

- [26] B. Reselman, "3 questions to answer when considering a multi-cluster Kubernetes architecture," 14 December 2021. [Online]. Available: <https://www.redhat.com/architect/multi-cluster-kubernetes-architecture>. [Accessed 15 December 2021].
- [27] T. Yuan, "A Multi-Cloud and Multi-Cluster Architecture with Kubernetes," 13 November 2019. [Online]. Available: https://www.alibabacloud.com/blog/a-multi-cloud-and-multi-cluster-architecture-with-kubernetes_595541. [Accessed 5 February 2022].
- [28] Improbable, "Introducing kEdge: a fresh approach to cross-cluster communication.," 1 January 2018. [Online]. Available: <https://www.improbable.io/blog/introducing-kedge-a-fresh-approach-to-cross-cluster-communication>. [Accessed 5 February 2022].
- [29] L. Osmani, "Multi-Cloud Connectivity for Kubernetes in 5G Networks," *IEEE Communications Magazine*, vol. 59, no. 10, pp. 42-47, 2021.
- [30] WireGuard, "WireGuard Installation," [Online]. Available: <https://www.wireguard.com/install/>. [Accessed 1 March 2022].
- [31] Network Service Mesh, "Test kernel to wireguard to kernel connection," [Online]. Available: <https://github.com/networkservicemesh/deployments-k8s/tree/main/examples/use-cases/Kernel2Wireguard2Kernel>. [Accessed 11 March 2022].
- [32] Linkerd, "Linkerd Multicluster Manifests," [Online]. Available: <https://github.com/linkerd/website/tree/main/multicluster>. [Accessed 5 March 2022].
- [33] GCP, "Online Boutique – Microservices Demo," [Online]. Available: <https://github.com/GoogleCloudPlatform/microservices-demo>. [Accessed 10 March 2022].
- [34] Open Source, "Admiralty," [Online]. Available: <https://github.com/admiraltyio/admiralty>. [Accessed 15 March 2022].
- [35] Open Source, "Shipper," [Online]. Available: <https://github.com/bookingcom/shipper>. [Accessed 15 March 2022].
- [36] Open Source, "Kubernetes Cluster Federation," [Online]. Available: <https://github.com/kubernetes-sigs/kubefed>. [Accessed 15 March 2022].
- [37] Open Source, "mck8s: Container orchestrator for multi-cluster Kubernetes," [Online]. Available: <https://github.com/moule3053/mck8s>. [Accessed 15 March 2022].
- [38] <https://www.k8gb.io/>, "K8GB – Kubernetes Global Balancer," [Online]. Available: <https://www.k8gb.io/>. [Accessed 11 March 2022].
- [39] Liqo, "[Feature] Move PersistentVolumes between clusters," 27 January 2022. [Online]. Available: <https://github.com/liqotech/liqo/issues/1071>. [Accessed 10 March 2022].

- [40] Open Source, "Kubernetes External Secrets Operator," [Online]. Available: <https://github.com/external-secrets/external-secrets>. [Accessed 16 March 2022].
- [41] Observe Inc., "What Is Observability?," 16 February 2021. [Online]. Available: <https://www.observeinc.com/resources/what-is-observability/>. [Accessed 15 March 2022].
- [42] O. Hughes, "What is Gaia-X? A guide to Europe's cloud computing fight-back plan," 10 June 2020. [Online]. Available: <https://www.techrepublic.com/article/what-is-gaia-x-a-guide-to-europes-cloud-computing-fight-back-plan/>. [Accessed 5 January 2022].
- [43] C. Morris, "Infrastructure as a story," 2 September 2021. [Online]. Available: <https://www.thoughtworks.com/en-br/insights/blog/cloud/infrastructure-as-a-story>. [Accessed 11 January 2022].
- [44] Google, "Multi-cluster use cases," [Online]. Available: <https://cloud.google.com/anthos/multicluster-management/use-cases>. [Accessed 15 December 2021].
- [45] M. Caulfield, "KubeCon 2018," in *Clusters All the Way Down: Crazy Multi-cluster Topologies*, Seattle, 2018.
- [46] N. Leiva, "Kubernetes multi-cluster networking made simple," 19 December 2018. [Online]. Available: <https://itnext.io/kubernetes-multi-cluster-networking-made-simple-c8f26827813>. [Accessed 10 February 2022].
- [47] Platform9, "Difference Between multi-cluster, multi-master, multi-tenant & federated Kubernetes," 27 May 2020. [Online]. Available: <https://platform9.com/blog/difference-between-multi-cluster-multi-master-multi-tenant-federated-kubernetes/>. [Accessed 6 February 2022].
- [48] D. Friedlander, "Kubernetes-ClusterIP-Service," 24 September 2019. [Online]. Available: <https://www.docker.com/blog/designing-your-first-application-kubernetes-communication-services-part3/kubernetes-clusterip-service/>. [Accessed 6 February 2022].
- [49] V. Sharma, "Load Balancer Service type for Kubernetes," 8 August 2020. [Online]. Available: <https://medium.com/avmconsulting-blog/external-ip-service-type-for-kubernetes-ec2073ef5442>. [Accessed 6 February 2022].
- [50] G. Chandra, "Kubernetes Multi-Cloud and Multi-Cluster Connectivity with Submariner," 20 February 2020. [Online]. Available: <https://itnext.io/kubernetes-multi-cloud-and-multi-cluster-connectivity-with-submariner-test-cockroachdb-b79662209bd7>. [Accessed 20 January 2022].

Versions

Version	Date	Changes	Review summary
v0.1	22-12-2021	<ul style="list-style-type: none"> Document created Initial Structure Stakeholder Analysis (80%) Context Analysis (20%) 	<ul style="list-style-type: none"> Research questions should be used as a basis for structuring the report. Methods such as 7S, PESTEL are too heavy and not suitable for this project, they will not be used. Stakeholder Analysis needs a second look – in this project, it should be more about people who can help reach the goal.
v0.2	19-01-2022	<ul style="list-style-type: none"> Added Problem Analysis, User Stories, Prototype Log Improved structure 	<ul style="list-style-type: none"> Problem Analysis should be split in 2 parts – initial and technical deep-dive Conceptual Design should be described before the Technical Design
v0.3	03-02-2022	<ul style="list-style-type: none"> Changed the structure Added partial Technical Research 	<ul style="list-style-type: none"> Multi-criteria analysis adding weights and scores Move all appendices into a separate document
v0.4	23-02-2022	<ul style="list-style-type: none"> Elaborated Design Principles Added MCA results Improved Conceptual Design based on MCA Moved appendices to a separate document 	<ul style="list-style-type: none"> Summarize Stakeholder Analysis briefly in the main report and provide references to the full version Provide an explanation of the Conceptual & Technical Designs based on a single selected requirement; refer to the Appendix for the rest
v0.5	16-03-2022	<ul style="list-style-type: none"> Added Technical Design Added Advice Improved structure and formatting 	<ul style="list-style-type: none"> Elaborate further on why 3 different architectures are proposed
v0.6 Concept	20-03-2022	<ul style="list-style-type: none"> Elaborated on 3 different architectures (DP-1) Extended Advice 	<ul style="list-style-type: none"> Reduce main content down to 60 pages
v1.0 Final	03-04-2022	<ul style="list-style-type: none"> Content reduced Added conclusion and reflections 	N/A

Appendices

Appendices to this report can be found in a separate document – “Appendix to the Graduation Report”.