

Exploring the Benefits of Cloud Migration and Interoperability of Cloud Vendors for Small and Medium Enterprises



UNIVERSITY OF
LINCOLN

Dylan Samuel Petty
PET19703357

19703357@students.lincoln.ac.uk

School of Computer Science
College of Science
University of Lincoln

Submitted in partial fulfilment of the requirements for the
Degree of BSc(Hons) Computer Science

Supervisor: Dr. Derek Foster

May 2023

Acknowledgements

Firstly, I would like to express my gratitude to Dr. Derek Foster for fulfilling the role as project supervisor by inspiring me to take this research project and by sharing his expertise on the field of Cloud Computing.

Secondly, I would like to thank my friend Simon Jackson for sharing his vast networking prowess to help shape the foundations of the implementation stage.

Thirdly, I would like to thank my peers for supporting me through the many ups and downs throughout the academic year, through thick and thin they've kept a smile on my face and helped me walk the right path.

Finally, I feel indebted to my parents Karen Sutton and Anton Petty, my brother Connor Petty and my grandparents for their unconditional love and support inside and outside of this project.

Abstract

In the following dissertation, the author intended to assess the impact that cloud computing has over the modern IT market in the realm of computational services. This project aims to create an evaluation of on-premises resource costs of a rising SME in a transitional state versus 2 cloud platform's remotely located alternatives in conjunction with an SME's development.

With the prevalence of technology and land-ownership costs increasing at an alarming rate, businesses find themselves being pushed to make an ultimatum; either adhere to the increasing costs and remain with what is familiar or seek more logical solutions to meet these advancements head-on and reclaim physical space from once infrastructure hardware.

In this dissertation an infrastructure-as-code solution using HashiCorp provisioned configuration language, Terraform, has been implemented to provide insight into modern and more scalable infrastructure mediums. The results were validated against existing cloud features and examples to guarantee a reliable degree of accuracy.

This research found that cloud-based solutions provide not only diversity in payment methods for remotely sourced infrastructure and free up physical and human resources; but also develop understanding of the flexibility of cloud systems to relieve doubts about migrating to the cloud.

This project will utilise both Microsoft Azure and Google Cloud Platform throughout.

Key words: Cloud Computing, Terraform, Infrastructure Migration, Interoperability.

Table of Contents

Introduction	1
Literature Review	3
2.1.1 Cloud Architecture.....	3
2.1.2 Infrastructure as Code.....	5
2.1.3 Feasibility	6
2.1.4 Future of Cloud Computing.....	11
2.2 Aims & Objectives	13
Requirements Analysis	15
3.1 Stakeholders and End-Users	15
3.2 Functional Requirements	16
3.3 Non-Functional Requirements.....	17
Design & Methodology	18
4.1 Project Management	18
4.1.1 Scrum Sprints.....	21
4.1.2 Project Plan.....	28
4.2 Risk Analysis	32
Implementation	34
Results & Discussion	47
Conclusion.....	49
References	50

Chapter 1

Introduction

For the sake of this document, it is important to note that Beacon Inc is hypothetical hardware and software business in a transitional stage of development, they have given the task of migrating on premises resources to a cloud computing-based solutions to manage their infrastructure and existing resources. These include one virtual network with 3 corresponding subnets, 2 web application virtual machines which are externally load balanced, 2 SQL virtual machines to be internally load balanced, an instance of both private and public storage with relevant cloud computing components and firewall settings.

Beacon Inc is an up-and-coming SME(Small and Medium Enterprise) that offers hardware and software systems to customers in the emergent market of proximity-based retail. In order to thrive in the tech industry, Beacon Inc sought out Cloud Computing technologies. Evidently, they have become a prevalent contender to SMEs' on-premises storage methods, providing a virtual environment to store and manage services on a corresponding level (Tak et al, 2011,1); outsourcing any hardware needs according to a relative payment plan. Beacon Inc has predicted that migrating all services to the Cloud will support growth in the long run and provide virtual infrastructure for their data and computational needs. As a result of these predictions, this project is dedicated to exploring the capabilities of cloud migration for Beacon Inc and developing a variety of analyses based on a cost per feature comparison and interoperability between cloud vendor services.

A large consideration for cloud migration is whether Infrastructure as a Service (IaaS) or Platform as a Service (PaaS) would best benefit the requirements. PaaS certainly has benefits of having control over network components and services alike, providing a Platform to develop and deploy an application. However, PaaS renounces the hardware configurations to another infrastructure level leaving a good portion of their control of utilities to a third party (Bhardwaj et al, 2010, 61-62). Whereas, IaaS allows for

administrative control over the network alongside routing and storage services. IaaS also is preferred in terms of the flexibility of payment plans as they are not bound by just a contract, but more so charged for the resources that you use (Bhardwaj et al, 2010, 62). Therefore, for this project, IaaS is the superior option as it gives a lot more control over Cost Management, Network Administration, Scaling and Automation of services and User Permissions on top of the PaaS responsibilities. Using IaaS also means that Terraform (by Hashicorp) can be employed as an IaC(Infrastructure as Code) tool to produce repeatable and reusable code which can create and destroy infrastructure resources alike. Being able to reuse and redeploy code will save a lot of time rehashing redundant files (Basher, 2019, 20). There are some limitations to migrating to the cloud which will be promptly discussed; issues may arise when trying to migrate non-scalable applications with a fixed resource demand, in this case, the application would be a waste of cloud compute capability. Another potential limitation is that with the added control over the cloud resources, necessitates the management of user permissions as each collaborator will have their own technical preferences (Azeemi et al, 2013, 738).

Ultimately, throughout the duration of this project, the benefits of cloud migration will be explored and analysed to provide a true understanding of the interoperability between a variety of cloud vendors' services. An Infrastructure as Code solution will be developed in accordance with Beacon Inc's predictions. (Petty, D, S. 2022. 1-2).

Chapter 2

Literature Review

2.1.1 Cloud Architecture

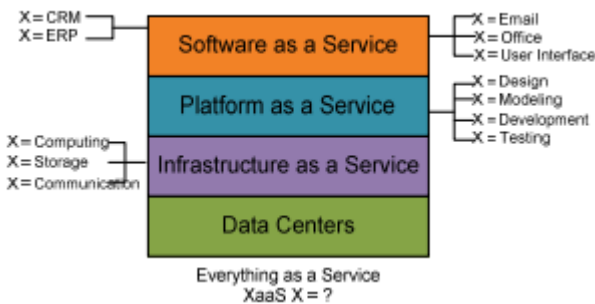


Figure. 1. Cloud Computing Hierarchy from Tsai et al (2010, 684)

Cloud Computing is a well-developed technological paradigm which soared has in popularity since its inception as shown by IT giants such as but not limited to Google, Microsoft and Amazon (Tsai et al. 2010. 684). Having individually adopted Cloud Computing, large corporations fought to create an industry leading platform for organisations to buy into for regular usage, iBeacon is a prime example. However, from Figure 1 above, only Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) are relevant to the scope of this project.

To understand the decision to use Infrastructure-as-a-Service (IaaS), it is important to make a comparison to its counterpart, Platform-as-a-Service (PaaS).

A large consideration Beacon INC have made is whether Infrastructure as a Service (IaaS) or Platform as a Service (PaaS) would be most suitable as their cloud adoption service. PaaS certainly has benefits of having control over network components and services alike, providing a Platform to develop and deploy an application. However, PaaS leaves the

hardware configurations up to another infrastructure leaving a good portion of their control of utilities to a third party (Bhardwaj et al, 2010, 61-62). Whereas IaaS allows for administrative control over the network as well as routing and storage services. IaaS also is preferred in terms of the flexibility of payment plans as they are not bound by just a contract, but more so charged for the resources that you use (Bhardwaj et al, 2010, 62). Therefore, for this project, IaaS is the superior option as it gives a lot more control over Cost Management, Network Administration, Scaling and Automation of services and User Permissions on top of the PaaS responsibilities. A potential downside to IaaS would be the need for technical expertise to set up and maintain Beacon INCs services. Using IaaS enables use of the IaC(Infrastructure as Code) program Terraform (by Hashicorp) can be utilised to produce repeatable and reusable code which can create and destroy infrastructure resources alike. Being able to reuse and redeploy code will save a lot of time rehashing redundant files (Basher, 2019, 20). In the scope of this project, Terraform will aid progress by allowing interoperability to be tested between a handful of cloud vendors, including Microsoft Azure and the Google Cloud Platform. The importance of this test will impact the decisions made by Beacon INC as should one cloud vendor prove more effective over another or a different department using separate vendors, then the Terraform templates can be reused; vendor lock-in is greatly combatted by this tool as compatibility issues are ever-lessening (Opara-Martins, J. 2016. 2-3).

2.1.2 Infrastructure as Code

Since Cloud Computing's inception, a console has been provided for platform-based configuration languages such as Azure's YAML (Sinha et al. 2000. 9-13) and Google Cloud's Jinja2 (Ronacher, A. 2008). Whilst these languages fulfil the configuration requirements for each cloud platform, difficulty ensues when the desire to create site to site connections across cloud vendors occurs; interoperability is a key practice used to ensure a work around of the vendor lock-in (Opara-Martins, J. 2016, 2-3). Infrastructure-as-Code (IaC) languages such as Terraform bridge the gap between vendor configuration jargon and multi-cloud infrastructures by collating participating vendor's services into one singular language, removing or reducing the need for platform specialists and creating a developer friendly language for ease of use (Morris, K. 2020).

Among many benefits that IaC provide, some key benefits are that the scripts can be reused by defining modules (Paloviita, O. 2022. 21) and written to dynamically deploy resources and firewalls; if there is an error with a setting, all the instances are deployed alike so its easier to isolate an oversight in comparison to other deployment methods (Basher, M. 2019. 17-19).

Another major benefit of using IaC solutions is that taking version control into account is very easy due to the fact that all of the code could be pushed to a github repository with comments stating changes to best manage infrastructure from a software developmental standpoint (Paloviita, O. 2022. 21).

2.1.3 Feasibility

The process of cloud migration has prevalent risks and rewards involved which will be discussed throughout this section from a business' standpoint. Firstly, the costs of said infrastructure would be the first field of questioning; especially for stakeholders with little background in IT. As Khajeh-Hosseini et al (2010, 452) mentioned in his methodology, the most effective direction for cost analyses on IaaS solutions is to compare all deployment and overhead costs in comparison to similar infrastructure on site. Khajeh-Hosseini (2010, 452) also provided a cost comparison between their anonymized company and their cloud-based counterparts as seen in Figure 2.

Period	Amazon Server Instances			Cmpny B
	2 small	1 small + 1 large	2 large	
1 Month	£200	£390	£590	£620
1 Year	£2,400	£4,680	£7,080	£7,440
5 Years	£12,000	£23,400	£35,400	£37,200

Figure 2. Cost Comparison of AWS server instances vs on premises infrastructure

Another study by Johnson et al (2019, 899) in the medical field discovered that cloud infrastructure costs have been lower by quite the substantial margin for costs accumulated over a 2-year period. On top of lowered needs for staff managing hardware on-premises, cloud solutions also offer a decreased overhead cost as facility

management, hardware maintenance and energy costs are outsourced to any chosen cloud vendor(s).

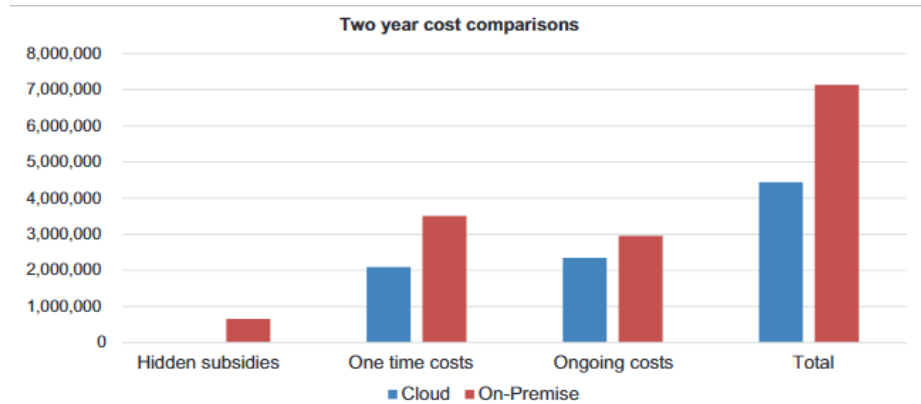


Figure 3. A graphical representation of Figure 3's stats from Johnson et al (2010, 899).
Values in \$USD

Cost Type	Importance	Cloud	On-Premise
Hidden subsidies			
Facilities and energy	High	0	239,908
Indirect costs (grants)	Low	0	0
Disaster recovery system (readiness)	High	0	200,000
Business continuity (response)	High	0	100,000
One-time costs			
Customization/business process reconfiguration	High	0	150,000
System re-architecture	Medium	0	100,000
Integration	High	200,000	300,000
Migration	Low	150,000	50,000
Hardware	High	7,500	35,000
Software	Medium	271,580	0
Training	Medium	50,000	25,000
Staffing	Low	890,000	1,383,760
Reporting	Low	696,800	930,280
Remote and mobile access	High	0	550,000
Ongoing costs			
Subscription/licensing	High	346,991	217,767
Software	High	0	0
Data ingress/egress	High	0	0
Service management staff	Medium	0	0
Application/database administrator staff	Medium	0	50,000
System administration staff	High	0	200,000
Security staff and related infrastructure	High	150,000	200,000
Technical support staff	High	800,000	800,000
Functional staff	High	1,150,000	1,150,000
Vendor/contract management staff	Medium	50,000	50,000
Total		4,762,871	6,731,715

Figure 4. A Statistical analysis from Johnson et al (2010, 899).
Values in \$USD

A final study to further validate the benefits of adopting cloud computing comes from a study by Doherty et al (2015, 514-515) who found that migration to the cloud can maximally utilize the resources acquired by a firm. An example of this would be freeing up employee resources by reducing the need to maintain as much hardware and partaking in other potentially non-core activities and move their attention to core skills or responsibilities instead.

Ultimately the decision to adopt the cloud is generally reliant on SMEs readiness to expand, expend resources and adhere to modern technological standards (Abdollahzadegan, A. 2013. 71-72). It is important to note that buying tech has no Return on Investment as it doesn't source any direct revenue but instead acts as a tool to acquire it. Contextually, Beacon INC have sought after an Infrastructure-as-a-Service solution so it can be assumed that the prior considerations have been made.

As mentioned above in section 2.1.2, a cost per feature analysis is a decisive factor into the feasibility of cloud migration as with technological advancements and financial uncertainties may alter the pricing of said services and alter the reality of adopting cloud computing into the firm. As seen in the study by Li et al (2009, 94-97) a lot of analyses based on costs such as employing software, energy and cooling costs are all prevalent factors in the process of reaching an informed decision to migrate on-premises infrastructure to cloud computing. This project will have a cost per feature analysis in later documentation (Petty, D, S. 2023. 1-4).

A key concern for cloud computing is ensuring data availability across a specified landscape, these landscapes consist of Local redundancy, Zonal redundancy and Geographical redundancy. The aforementioned plans are important for ensuring data integrity and fault tolerant storage among a cloud platform. Storage access tiers are another component that will alter the pricing as this takes into consideration how often the data is accessed; the most applicable access tiers to the scope of this project would be Hot and Cold access tiers, with hot being the more accessible of the two. In a study conducted by Daher et al (2018), the cost of access tiers on data accessibility for both Amazon Web Services and Microsoft Azure. See figures on the following page.

	AMAZON S3-Infrequent Access	Cool Blob LRS-Azure	Cool Blob GRS & RA-GRS Azure
Data Storage			
Amount of Data Stored	\$0.0125/ GB / month less than 30 days: Pro-rated storage charge.	\$0.01 / GB / month	RA-GRS: \$0.025/ GB / month GRS: \$0.02 / GB / month
Data Transfer			
Data write	\$0.0 Free	\$0.0025/ GB	\$0.005 / GB
Data retrieval	\$0.01 / GB	\$0.01 / GB	\$0.01 / GB
Transfer IN to cloud	\$0.0 Free	\$0.0 Free	\$0.0 Free
Transfer OUT to Internet	\$0.09 / GB	\$0.087 / GB	\$0.087 / GB
Transfer TO another region	\$0.020 / GB	\$0.087 / GB	\$0.087 / GB
Geo-Replication Data Transfer	Not Applicable	Not Applicable	\$0.020 / GB
Requests			
PUT	\$0.10 /10,000 request	\$0.10 /10,000 request	\$0.20 /10,000 request
POST (S3) / Create (Azure)	\$0.10 /10,000 request	\$0.10 /10,000 request	\$0.20 /10,000 request
COPY	\$0.10 /10,000 request	\$0.01 /10,000 request	\$0.01 /10,000 request
LIST	\$0.01 /10,000 request	\$0.10 /10,000 request	\$0.20 /10,000 request
GET	\$0.01 /10,000 request	\$0.01 /10,000 request	\$0.01 /10,000 request
DELETE	Free	Free	Free
Other requests	\$0.01 /10,000 request	\$0.01 /10,000 request	\$0.01 /10,000 request
Lifecycle transition requests	\$0.10 /10,000 request	Not Applicable	Not Applicable

Figure. 5. Amazon and Azure Cool access tier comparisons from Daher et al (2018, 88).

As seen in figure 5, the costs per request are very similar in most fields aside from Amazon's infrequent access having a lower cost for data write and cross regional transferring whereas Azures cool blob with Locally Redundant Storage has lower costing storage per GB, transferring to the internet costs and significantly lower data copying costs which may apply to copying data to redundancy storage.

Storage Amount	Storage Prices (US Dollars)	
	HOT TIER	COOL TIER
First 50 TB/month	LRS: \$0.0208 GRS: \$0.0458 RAGRS: \$0.0478	LRS: \$0.0152 GRS: \$0.0334 RAGRS: \$0.035
Next 450 TB/month	LRS: \$0.02 GRS: \$0.044 RAGRS: \$0.0459	LRS: \$0.0152 GRS: \$0.0334 RAGRS: \$0.035
Over 500 TB/month	LRS: \$0.0192 GRS: \$0.0422 RAGRS: \$0.044	LRS: \$0.0152 GRS: \$0.0334 RAGR: \$0.035

Figure 6. Azure blob storage pricing per GB from Daher et al (2018, 88)

Storage Amount	Storage Prices (US Dollars)		
	Standard Storage	Standard - Infrequent Access Storage	Glacier Storage
First 50 TB / month	\$0.026 per GB	\$0.019 per GB	\$0.005 per GB
Next 450 TB / month	\$0.025 per GB	\$0.019 per GB	\$0.005 per GB
Over 500 TB / month	\$0.024 per GB	\$0.019 per GB	\$0.005 per GB

Figure 7. AWS storage pricing per GB from Daher et al (2018, 88)

In figures 6 and 7, although not explicitly stated in the former the prices are calculated per GB of storage where the Cool and infrequent access storage tiers are most suitable to backups and large quantities of data are to be stored which aren't accessed regularly. The Glacier tier is more comparable to Azure's Cold access tier which is not listed in the above figures. Generally, AWS has lower overall costs for standard access tiers over Azure's Hot tier when using redundancy plans over Locally Redundant Storage. It is important to note that whilst Azure's Hot tier LRS plan is lower than AWS' Standard tier, the user may experience a phenomenon known as Vendor lock-in (Opara-Martins et al, 2016, 2-8) which states that whilst cloud adoption soars, businesses may become dependent on one platform's storage appliance as migrating large quantities of data to another platform can be very costly; hence the need for interoperability across cloud vendors to partially mitigate this issue. The main cause of vendor lock-in is usually a mixture of initial financial constraints when first migrating to cloud services and a short-range management perspective.

Wang et al (2013) found that for Amazon Web Services storage bucket S3s, there were two redundancy plans offered: Reduced redundancy plan ensuring 99.99% durability and the Standard redundancy ensuring 99.999999999% durability. It was found that users employing the latter storage plan would pay 20% more meaning that such plans to provide a small improvement to data redundancy may prove to be financially unavailable to smaller enterprises such as iBeacon Inc in its current state as deploying to more than just locally redundant storage may stretch their migration budget thin.

2.1.4 Future of Cloud Computing

The longevity of cloud computing as a standard is seemingly unquestionable as many organisations opt for cloud solutions for reasons stated in the above sections. It is important to note that for large enterprises there is far more leniency in terms of cloud infrastructure as resources are far more available in comparison to small and medium enterprises; Abdollahzadegan(2013,70) referred to this phenomenon as “Resource Poverty” which symbolises various constraints such as but not limited to financial strain, lack of professional personnel and short range business perspectives.

While migration risks are prevalent for large enterprises, the implications are far more devastating on a smaller enterprise as they have “less slack” with which to manage any unsuccessful investments, thus prolonging or thwarting any future successes. Despite these risks, surviving as a business in a heavily competitive environment demands changes in the very infrastructure of daily operation; adaptability, availability, scalability and security are of some major considerations for the longevity of a business on newly acquired cloud-based solutions.

Adaptability in the cloud computing environment is an extremely important factor as cloud infrastructure is ever changing to maintain highest hardware and software standards, this includes the very standard of resources currently utilised by many businesses alike; a close eye must be kept on cloud platform changes as hardware changes might require different security measures to continue operations as intended. Cloud engineers and architects in this scenario are required to follow new cloud infrastructure mediums to ensure the most efficient and cost-effective solutions for their clients (Kandukuri et al, 2009, 518-519). An important consideration for adaptability would also be that with the hardware of the data centre constantly changing, the security standards required by the new hardware may collide with existing configurations so a watchful eye must be employed to maintain a professional security standard.

Availability and Scalability go hand in hand as resources can be deployed dynamically to scale automatically when a threshold is met, for example when a certain workload is met (Lehrig et al, 2015, 87), the resource might scale vertically by employing more physical machinery such as Random Access Memory (RAM) to share the workload, or horizontal

scaling could be employed to add more virtual instances to spread the load. In the case of horizontal scaling, an example would be releasing more Virtual Machine (VM) instances rather than increasing the power of them (Ross et al, 2019, 329-330).

Cloud adoption evidently shares vast similarities between consumer needs and large-scale data availability in the context of many businesses. However, once Healthcare organisations come into play, the risks of data security ethics often outweigh even the benefits that cloud computing can supply. As found by Mehrtak et al (2021), although cloud computing can reduce the infrastructural and operational in-house costs and greatly increase the availability of data between consumers, such as physician and patient, this incredible benefit comes at the cost of data security ethics. A concern of paramount importance would be that the physical location of extremely sensitive data is unknown to the organisation that created it; ethically this is unjust and logically has given unauthorised personnel access to the sensitive data. Al-Issa et al (2019, 4) found that in conjunction with prior discussion, delegating data control to the cloud provider leaves the data more susceptible to data breaches. It was also mentioned that with an increased number of devices and points of contact the risks of data compromise increases in parallel with extra angles of attack from a potential Threat Actor. Furthermore, it is a necessity that the patient can trust the organisation that their data is handled with utmost care and consideration. For an organisation as such to survive technologically, it must employ only capable industry specialists to ensure professional standards are thoroughly enforced and a concurrent audit is upheld (Doelitzscher et al. 2012. 378-380).

2.2 Aims & Objectives

This project will be split into two primary aims.

1. To explore the capabilities of cloud computing and research developmental approaches for Terraform.
2. To implement and test Terraform solutions that meet the requirements of Beacon Inc.

The first aim focusses on a research-based approach to bridge the gap between the foundations and required knowledge to complete this project and an implementation-based approach to document the steps which have been taken throughout the process of this project.

1. Research
 - a. To complete at least 6 CloudSkillsBoost lab exercises by the end of December 2022, and evaluate the tested resources suitability for the proposed implementation.
 - b. To create a significant technical diagram by early January 2023 mapping out the network topology of both current on-premises network and initial proposed virtual network topology.
 - c. Design an IP configuration for all of the network appliances by late January 2023 so devices and functionality can be mapped effectively.
2. Implementation
 - a. To decide which cloud platforms are most feasible for this project by early February 2023 to decide on payment plans.
 - b. To develop relevant networking resources by midway through February 2023 in order to correctly plan out resources around them e.g. subnets
 - c. To develop virtual machine instances by late midway through March 2023 so that resources can be deployed to them and site to site connections can be tested.

- d. To configure other network devices for each virtual machine instance by late march 2023 and ensure that all relevant configurations have been met.
- e. To test site to site vpn connections by early may 2023 to ensure that some sense of interoperability has been tested for the conclusion.

Chapter 3

Requirements Analysis

This section is dedicated to identifying the stakeholders and end users will be addressed, alongside a set of functional and non-functional requirements to be defined.

3.1 Stakeholders and End-Users

Beacon Inc has many points of contact in which will have interest and utilise the interactivity of the infrastructural components whether through an RDP connection to a vm or through a Software-as-a-Service (SaaS) layer program (Dubey et al. 2007. 2-5) ; these figures represent the needs and wants of an industry leading SME.

1. Shareholders of Beacon Inc will require updates on the infrastructure as to maintain trustworthiness with their clients; in the event of a failed cloud adoption, shareholders may seek to sell their shares and move funds to other rivalling businesses, this would be catastrophic for an SME in a transitional stage of growth.
2. Executives of Beacon Inc will only employ the highest standard solutions for their cloud adoption as a failed adoption may lead to loss of trust, revenue and even data if the adoption was semi completed and reverted without a proper risk assessment involved. (Cayirci et al. 2014. 2-5)
3. Beacon Inc's cloud engineers beyond the deadline of this project must be able to pick up the IaC solutions and potentially adapt them for newer technological mediums to suit their needs.
4. An end user of Beacon Inc would be the customers who would buy hardware or software through the business as usual through a form of web-based application hosted through the virtual machines requested. These virtual machines may host sites like PCBuilder for dynamically configuring a purchase; this process queries a lot of data if the customer is uncertain, meaning that to save computing resources the virtual machines must be configured for a fast but feasible medium.

Implementation objectives are broken down into functional requirements for capabilities of the IaC solutions and non-functional requirements used to describe how well the solution performs said capabilities. Additionally, ISO standard 29148 will be followed as a specific ruleset is enabled for quality assurance of the requirements, specifically it states that they must be (Garcia et al. 2020. 3):

- Validated by a stakeholder of whom has interest in the system throughout its lifecycle.
- Define constraints for stakeholder requirements which are unavoidable.
- Define requirements in a way that won't undermine the project if changed.

3.2 Functional Requirements

1. The IaC solutions must use correct syntax with relevant parameters provided otherwise potential functionality is lost from omitting data.
2. The solution must deploy at least 1 network with 3 subnets, each with their own dedicated CIDR ranges with suitable IP availability.
3. At least 1 virtual machine must be deployed upon running a solution with relevant parameters so at least a portion of functionality is met.

3.3 Non-Functional Requirements

1. The structure of groupings should be in a hierarchical format with one overarching management group to other management environments to subscriptions and finally cascading down to resource groups, as seen below in figure 8.
2. “Terraform fmt” must be used to ensure code readability by neatly aligning resources.
3. IP ranges stay consistent throughout the project.
4. Dynamic assignment of resources should be used to ensure professional code practices and reusability.

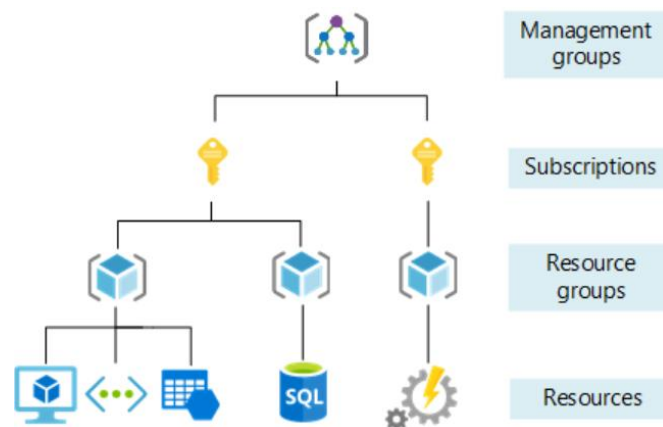


Figure 8. Azure hierarchy from [learn.microsoft/organise-resources](https://learn.microsoft.com/en-us/azure/management/organise-resources)

Chapter 4

Design & Methodology

4.1 Project Management

The Agile project methodology sets some key principles in order to enable high flexibility with continuous short ranged and faster paced than its more generally used counterpart, The Waterfall methodology. This is especially the case with software development principles as resource collisions may occur at any point during the project management lifecycle which will require a break in the waterfall structure; the shorter iterations of the lifecycle components make it ideal for this project as debugging can takes an indefinite amount of time as quantifying time required for a bug fix in an active workspace is near impossible and can only be led by predictions or assumptions otherwise (Thesing et al. 2021).

The Agile methodology (Beck et al. 2001) is comprised of 12 key principles which help maintain integrity of the project at hand in many ways, some of which are applicable to this project:

- Working software is the primary measure of success.
- Welcome changing requirements no matter the stage.
- Regularly reflect on effectivity towards a solution

Another contender for most suitable methodology would be the Scrum Agile framework which whilst very similar to the agile methodology, it hones on delivering timely changes to ensure that schedules are kept tight and products are quality assured regularly to produce a high value output (Darwish et al. 2016 24-25).

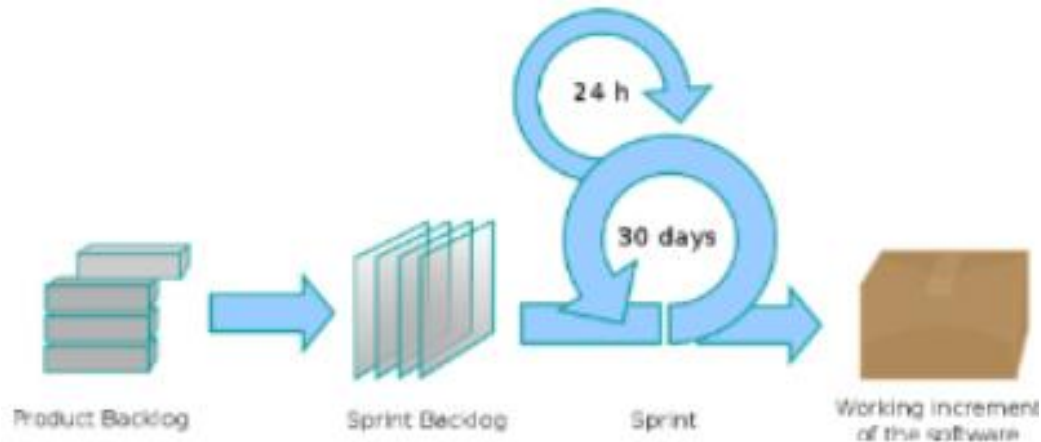


Figure 9. Scrum Methodology visualisation from Lei et al (2017)

As seen in Figure 9, the scrum methodology is comprised of a product backlog where requirements, functions and enhancements of a product are split into manageable tasks by the Product Owner of the management team. Then these requirements get processed into sprint backlogs which the team clarifies what functionality is required by the end of the sprint to ensure the schedule is followed thoroughly. If the team realises that more work on a task is required then the task is added back into the Product Backlog with updated requirements (Lei et al. 2017, 61).

As seen in the project task board below in figure 10, The Kanban methodology utilises a Just-In-Time delivery by defining tasks by what needs to be done with a tightly constrained deadline. This is utilised by colour coding tasks in terms of importance or relevance to the overall goal, meaning that the core components aren't easily pushed aside. These principles are enforced by a following a short list of core values, some of which consist of:

- Limiting the number of items allowed into the Work In Progress stage to streamline progress and ensure nothing is getting skipped.
- Increasing throughput by assigning more human resources to relevant tasks.
- Creating a visual representation of the work progress so that its easily seen and understood without having to open any documents; potentially the first thing you see when entering an office.

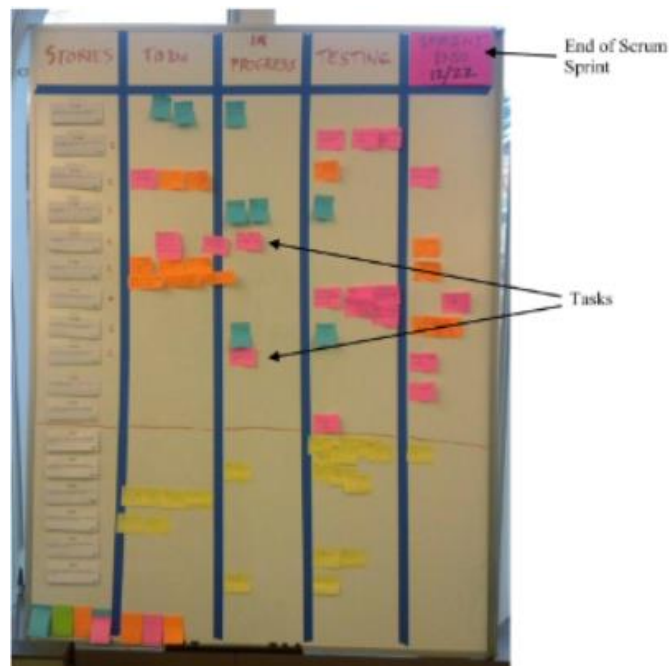


Figure 10. Kanban Visualised task board from Lei et al(2017)

While Scrum is more suitable for this project than its adapted counterpart due to the fixation on sticking to a schedule over enabling an indefinite number of incremental stages, Kanban is a project methodology which focusses on visualising the workflow which would be far more applicable if this project were to have a product management team rather than just one Cloud Adoption specialist; therefore, Scrum is the dedicated project methodology.

4.1.1 Scrum Sprints

Scrum in essence starts out with a vision; however vague it may be a baseline of cost and time is set to lock in the duration of the project and ensure a finished product by the end (Schwaber et al. 2010. 2). Given that there's only one developer in the team, they have all control over the delegation of tasks and lengths of sprints. This project was conceived with a basic understanding of cloud computing and networking concepts and no real concept of working hours until after the first sprint. Each sprint is comprised of 3 iterative stages, the findings of which will be collated in each section: these being Design, Implementation and Testing.

Sprint 1: consisted of mainly collating the research and planning stages defined in the aims and objectives which took place in the timeframe of 30 days from January to late February 2023. This sprint was dedicated to procuring resources necessary for this project to function such as a significant technical diagram with corresponding network devices and logical structure as seen in the figure 11 below. This logical structure will alter be changed in accordance to scrum rules and change of perspective conceptually.

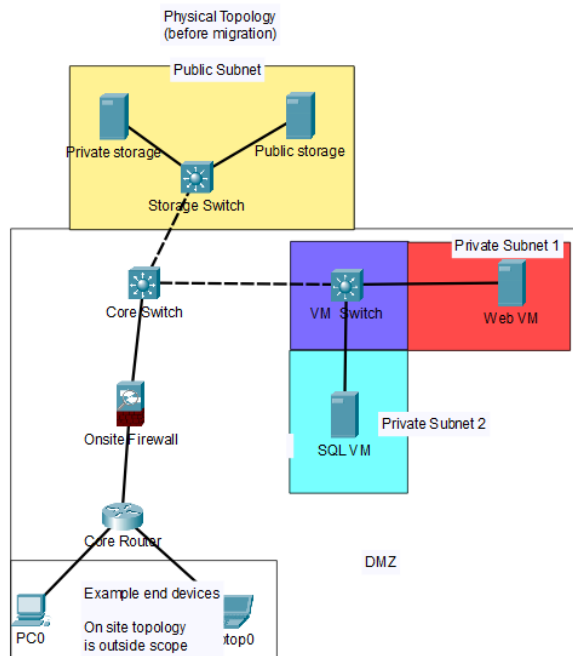


Figure 11. On-premises topology before cloud migration from Petty, D, S (2023)

As seen in figure 11, the on-premises network is basic as on-premises resources remain limited due to cost constraints for physical hardware and a total of up to 249 potential end user devices as stated by the medium enterprise classification size. (gov.uk . 2023)

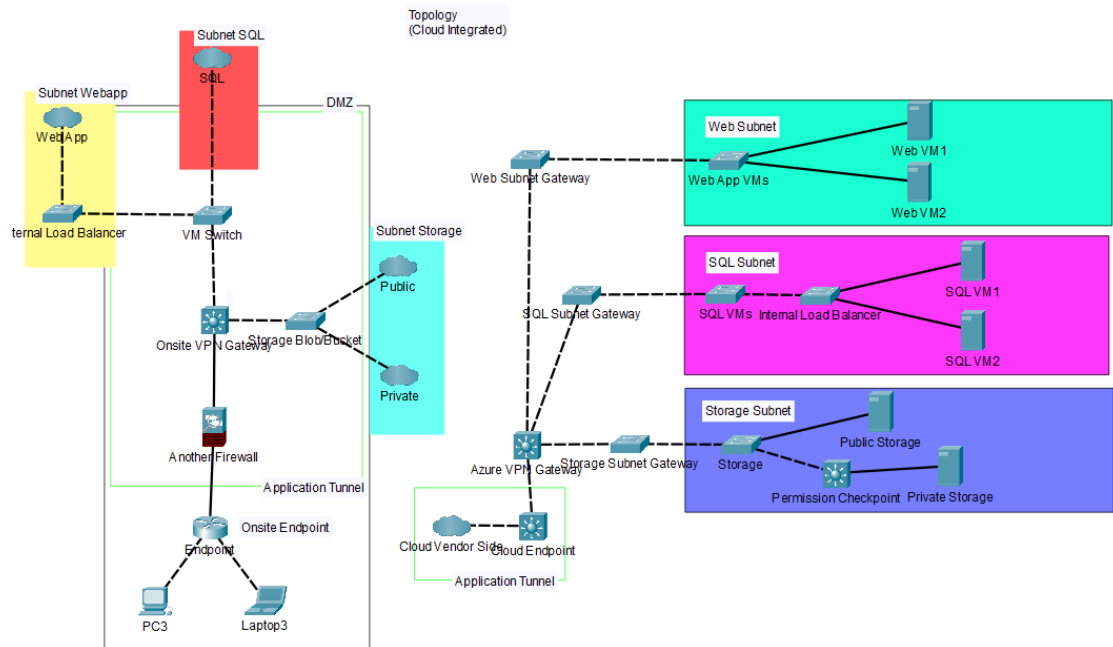


Figure 12. Initial cloud integrated topology from first sprint. Petty, D, S (2023)

Upon completion of these designs, a basic virtual network structure was implemented into a Terraform script to help develop fluency with the programming language and the knowledge required to succeed throughout the project. These steps are of paramount importance as if an incorrect structure is implemented, the remaining scrum sprints will be more tightly constrained against the timeframe while patching up mistakes from earlier. While the general network topology was functional, in order to develop a more sustainable solution this topology was reworked later on.

To validate the understanding of the required networking concepts for the project, the results of the terraform network deployments helped to draw conclusions to the first sprint in terms of best practice for scalability and what functionality could be utilised with a working terraform solution.

Sprint 2: This sprint lasted from late February to late March in which the foundations of the implementation stage were to be put forward; Terraform resources were planned to be collated and logically turned into IaC components and thoroughly tested along the way. Two Major constraints would hinder this sprint such as core knowledge labs from CloudSkillsBoost have been disabled since the beginning of this project. See Figure 13 for context.

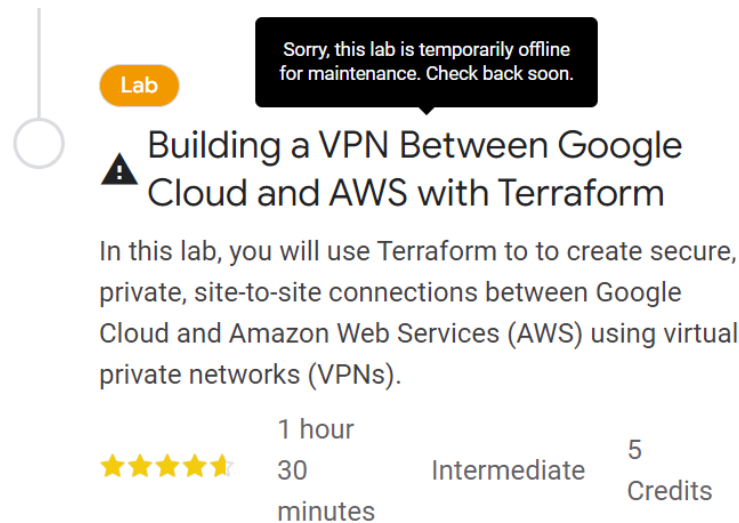


Figure 13. CloudSkillsBoost “managing infrastructure with terraform” questline blocked from maintenance.

A direct result of this constraint meant that because this fundamental resource that was relied upon for during the planning phase of the project, time that would’ve been spent on implementation and testing had then been redistributed to self-teaching which greatly lessened the manoeuvrability of task allocations and the ability to produce a fully functional solution on time after proper mitigation practices.

An unforeseen constraint from this sprint was that a potentially life-threatening health related issue appeared leaving the developer extremely anxious and unable to work at peak performance for around a month. The only mitigation for this circumstance is that the extra contingency time which is given to longer tasks for unlikely events as such. This did no favours to the project life cycle after the first interruption so extra contingency plans were exhausted and final terraform solution functionality became limited. A conclusion ready for the final sprint was drawn in that regardless of the

functionality from the IaC solution at the final deadline, a solution using Azure and Google Cloud Platform's GUIs will be implemented as a final measure to ensure that Beacon Inc at least have a functional product befitting of their requirements.

Testing for this sprint was concise and timely as a full deployment would take several minutes so smaller iterations of the project were created in conjunction with the scrum methodology to test smaller instances of the terraform code to ensure that each part is working as intended. These results were then collated into a local directory to enable version control of working prototypes for best practice of software engineering in accordance with Agile methodology principles. (Larman, C. 2004)

Sprint 3: The third and final sprint is dedicated to the final implementation objectives in which draws this last Scrum sprint close to the final deadline for the project which leads to the cloud solutions hand over. The duration of this scrum sprint was from early April until the middle May which is a relevant timeframe for polishing the solution for hand over. As a follow-on effect from the previous two constraints, the complexity of the terraform solution has decreased however an alternative approach to reach the same outcome will be discussed in the implementation chapter after this one.

During this stage the topology of the proposed network would be redesigned to best align the network with industry standard practices, this is because the earlier stages proposed a less scalable solution due to the fact that it doesn't consider the need for a separate subnet for the VPN gateway to connect from site to site. Relevant firewall settings are employed to ensure that only specific connections come through verified and controlled ports for maximum integrity of the network and protection against Threat Actors. An implementation of this would be a Demilitarised Zone (DMZ), which Jack Webb (2014) describes as a designated area on a network which is untouched by unauthorised users which maintains integrity by only interacting with trusted networked users. Another level to this would be utilising user policies applied to corresponding user groups, these could range from being assigned on a departmental basis to one which follows the Principle of Least Privilege (PoLP).

Claycomb (2012) conducted a study on the possibility of an insider attack on a cloud-based network, three main types of threats were identified which consist of:

- **Rogue Administrator:** A Threat Actor who was employed to a cloud provider with malicious intent will commonly result in theft of sensitive information which leaves the targeted company with lessened data integrity and a breach of confidentiality. This threat can vary in severity depending on the business that's targeted; the most sensitive data to be stolen is medical records as the healthcare provider must uphold confidentiality or crimes could potentially be committed by using the stolen information (Chernyshev et al. 2019).
- **Cloud Exploiter:** A cloud-related insider threat that Claycomb (2012) describes as an inconsistency with security policies which could lead to unauthorised access to sensitive information or organisation systems. This type of threat can be either maliciously intended or accidental as this vulnerability is often caused by differences in policies from cloud and local systems. This type of threat may also occur when data centre hardware is upgraded as new compatible firmware may require updated firewall rules. This attack has been found to be used mostly for stealing sensitive data to sell or stealing intellectual property.
- **Nefarious Activity:** A Threat Actor who attempts to use cloud services to enact an attack against their employer, these kinds of attacks include but are not limited to:
 1. Using Cloud processing power to crack password blocked files allowing unauthorised access.
 2. Using relatively cheap systems to launch a Distributed Denial of Service attack on the business systems.
 3. An insider leveraging sensitive company information like trade secrets and taking them to a job in a competitor business.

These attacks although seemingly uncommon, can be hard to detect, thus proving the importance of a tight grip on role-based access control (RBAC) configuration. The Principle of Least Privilege is a determining ruleset when defining access roles, as it states that permissions must be granted for no more than what is necessary for their job responsibilities (Schneider, F.B. 2003).

While Role-Based Access Control (RBAC) initially would have been implemented as an exemplar case for Beacon Inc when applying roles. However, due to various time constraints, this consideration has been removed from the agenda.

In this case it would be the web and SQL subnets as inbound network traffic from outside connections is managed internally from the gateway subnet with a combination of firewall rules and private IP routing. This is possible through the use of BGP peering as once a network gateway is setup with a vpn tunnel, it will learn connected IP addresses by exchanging routing information defined by their Autonomous System Numbers (ASN's). BGP peering is especially useful in a large and scaling network as this method automatically exchanges information with reachable IP addresses between two BGP peers. (Huston et al. 2010)

This peering method has been tested in the final sprint before the final deadline and is able to detect the IP addresses of subnets in both platform's local networks. See figure 14 below for evidence.

BGP peers

Peer address	Local address	Asn	Status	Connected duration	Routes received	Messages sent
10.2.0.4	10.2.0.4	65002	Unknown	-	0	0
10.2.0.4	10.2.0.5	65002	Connected	00:50:34.1199366	0	71
10.2.0.5	10.2.0.4	65002	Connected	00:50:34.1164997	0	71
10.2.0.5	10.2.0.5	65002	Unknown	-	0	0
169.254.21.2	10.2.0.4	65003	Connected	00:21:11.4432134	3	42
169.254.22.2	10.2.0.5	65003	Connected	00:14:48.0139211	3	29
169.254.22.2	10.2.0.4	65003	Connecting	-	0	0
169.254.21.2	10.2.0.5	65003	Connecting	-	0	0

Showing only top 50 BGP learned routes in the grid, click Download learned routes above to see all.

Learned Routes

Network	Next hop	Local address	Weight	Source peer	Origin	AS path
10.2.0.0/16	-	10.2.0.4	32768	10.2.0.4	Network	-
10.3.0.0/16	169.254.21.2	10.2.0.4	32768	169.254.21.2	EBgp	65003
10.2.0.0/16	-	10.2.0.5	32768	10.2.0.5	Network	-
10.3.0.0/16	169.254.22.2	10.2.0.5	32768	169.254.22.2	EBgp	65003

Figure 14. Evidence of BGP peering table from Azure Portal.

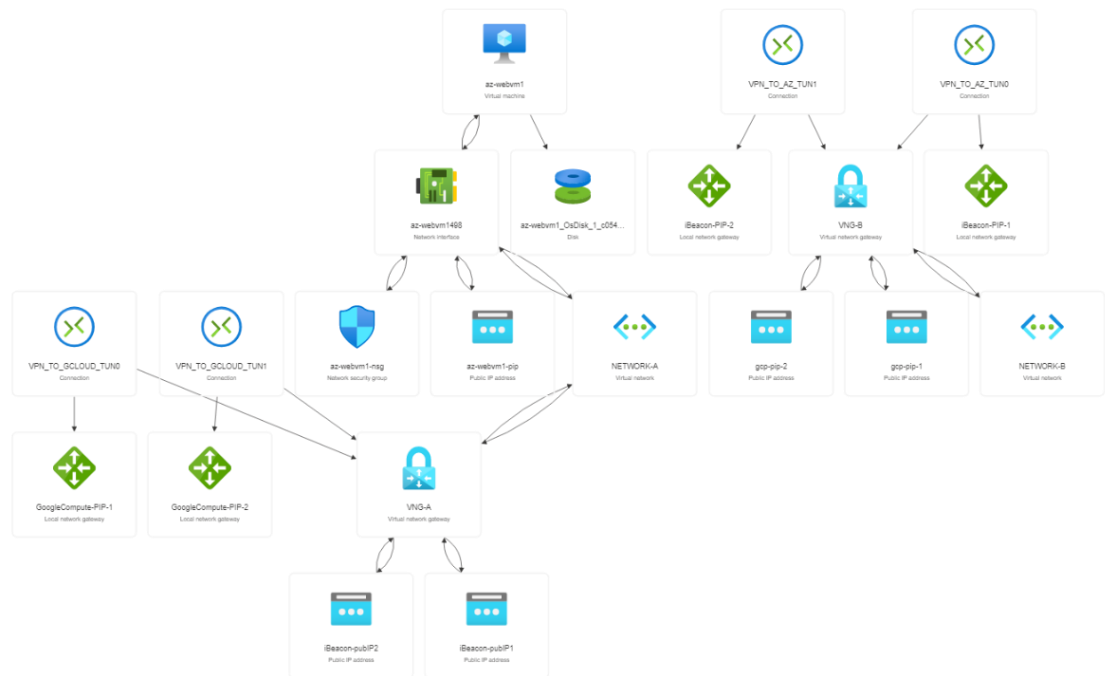


Figure 15. UML dependency diagram, origin discussed later in the document.

Figure 15 illustrates the ideal resource connectivity by following professional practices for utilising network appliance functionality to ensure the creation of a high standard cloud adoption network for Beacon Inc to adopt as a permanent working environment. This was an emergency iteration that was discussed during a progress meeting that the old topology was ineffective for implementing a vpn gateway with relevant routing configuration.

4.1.2 Project Plan

As a follow up from 4.1.1, a simple Gantt chart has been created in an earlier document which loosely describes a timeframe for each task which has been split into various scrum sprints according to the objectives. (Petty, D,S. 2022)

While the codes for the objectives don't directly correlate to the tasks undertaken in the sprints, this Gantt chart is only fully applicable to projects that follow the Waterfall methodology and should be interpreted as a general guide for the workflow.

See figures 16 and 17 below for colour and code key.

Colour Key	
Estimated Time	
Contingency	

Figure 16. Gantt chart colour key. (Petty, D,S. 2022)

Objective Key:									
Objective:	Code:								
1	A	Develop an understanding of the most suitable project methodologies							
	B	Explore various Cloud vendors like Microsoft Azure or Amazon Web Services							
	C	Write up a Project Proposal and Submit an Ethics form							
2	D	Familiarise myself with Cisco Packet Tracer							
	E	Create a significant technical diagram of the intended cloud systems							
3	F	Familiarise myself with the services necessary							
	G	Learn how to use Terraform and document benefits							
4	H	Start implementing Terraform solutions							
	I	Test half way to ensure correct implementation							
	J	Complete development of Terraform templates							
	K	Final test and polish up the code as final artefact piece							
5	L	Collate findings for next objective							
	M	Use findings to compare pricing of each service							
	N	Create a cost/feature matrix							
	O	Finalise analysis documentation							

Figure 17. Objective key for Gantt Chart. (Petty, D,S. 2022)

Sprint 0:

The first section of the Gantt chart is dedicated to the research stages defined in the earlier parts of this document which are better completed in accordance with the Waterfall methodology as the linear and methodical approach to these tasks makes it prevail over the Agile methodology in this instance. (Aroral, H, K. 2021)

See figure 18 for the timeframe.

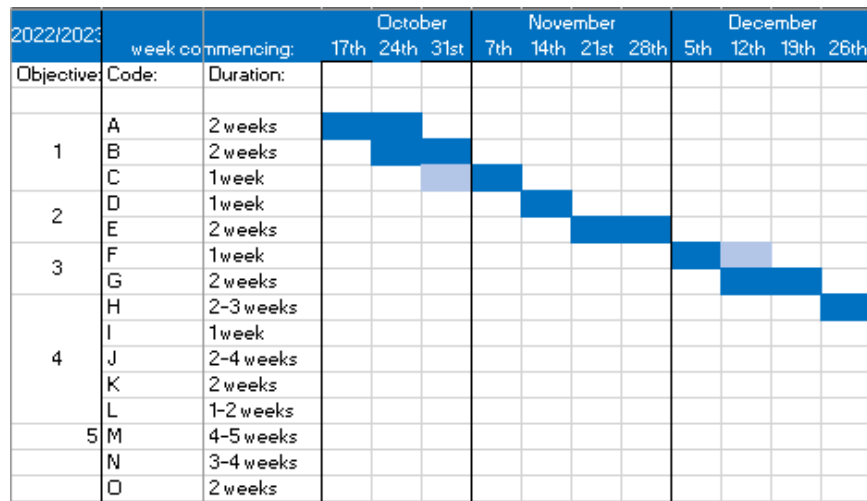


Figure 18. Research objectives of the project (Petty, D,S. 2022)

Sprint 1:

The first sprint of the project is where the first iterations of the terraform solution will be implemented including network foundations and relevant IP addresses for all initial components, including subnets and CIDR ranges. See Figure 19 for a visualised timeframe.

2022/2023	week co	January					February		
		2nd	9th	16th	23rd	30th	6th	13th	20th
Objective:	Code:								
1	A								
	B								
	C								
2	D								
	E								
3	F								
	G								
4	H								
	I								
	J								
	K								
	L								
5	M								
	N								
	O								

Figure 19. First sprint timeframe (Petty, D,S. 2022)

Sprint 2:

The second sprint consists of getting most of the core functionality of the solution finished in a 30-day timeframe as most of the groundwork has been set and resources gathered from prior sprints, that the majority of software progress should come from here. The contingency time for the first sprint was exhausted following the two major constraints discussed earlier. See figure 20 for the second sprint timeframe.

2022/2023	week co	March				
		27th	6th	13th	20th	27th
Objective:	Code:					
1	A					
	B					
	C					
2	D					
	E					
3	F					
	G					
4	H					
	I					
	J					
	K					
	L					
5	M					
	N					
	O					

Figure 20. Second sprint timeframe (Petty, D,S. 2022)

Sprint 3:

The final sprint of the project consisted of generating a cost per feature analysis and collating all infrastructure iterations into one final product to hand over. A couple compromises had to be made due to the looming deadline and the setbacks faced in the previous sprints, these being a lowered IaC solution however a cloud-based solution will be offered regardless of complexity. See figure 21 for final sprint timeframe.

2022/2023	week co	April				May	
		3th	10th	17th	24th	1st	8th
Objective:	Code:						
1	A						
	B						
	C						
2	D						
	E						
3	F						
	G						
4	H						
	I						
	J						
	K						
5	L						
	M						
	N						
	O						

Figure 21. Final Sprint Timeframe (Petty, D,S. 2022)

Any further detail on the final solution will be provided in the coming sections.

4.2 Risk Analysis

Due to the ever-changing nature of the Agile and Scrum methodologies and the flexibility they explore, often complications arise causing potential setbacks for the project as a whole. Ward Cunningham (1992) introduces a concept technological debt which states that poorly designed software can create a large backlog of problems further down the line. A little debt would speed up the development, but cumulative unpaid debts start to create larger problems. Therefore, each sprint cycle will be checked for technical debt before being paid off.

The table below displays potential risks of the whole project, followed by associated mitigation practices to handle the situation. Some of these risks have been partially discussed in above sections.

Risk	Implication	Mitigation	Severity
Inadequate learning resources	Time spent finding extra resources not used on implementation	Spend time between or before sprints to polish knowledge for most time efficient implementation.	Low
The developer is burned out from over working	Creates resource debts and loss of progress pace.	Utilise Scrum iterations to create smaller manageable work blocks	High
Approach to task is ineffective or inefficient	Time wasted on tasks that are wrong	Utilise Agile practices to respond to change and hold a meeting to discuss future.	Med
Run out of cloud credits	Resources cannot be deployed or tested	Set up a budget ahead of time and only deploy smaller iterations for testing to save budget.	High

Staff are unwell and unable to work	No progress as this project has one developer	Set aside extra time in planning for a contingency plan and enable working from home	Med
Sprint runs over time allocation	Task requirements to be reassessed	Utilise contingency time to reiterate the task with updated requirements	Low
Project runs over deadline	Client unhappy and lack of project plan	Utilise version control throughout the project to ensure that at least a working prototype is handed over.	High
Learning resources under maintenance	Slower uptake of required information and practices	Allocate more time to self-learning, less time for implementation	Med
Internet outages	Can't access cloud platform	Use mobile hotspot or talk to ISP about outages.	Low
Platform maintenance	Cloud services down	Use available resources or use time to plan ahead.	high

Table 1. Risk Matrix

Chapter 5

Implementation

5.1 Terraform Deployment

For the sake of not repeating content, in the event that two processes are the same on both cloud platforms, only one set of screenshots will be provided.

Before any progress can be made on the solution itself, the IP configuration of networking components has to be decided beforehand. See below a table that displays the corresponding IP addresses for the Microsoft Azure side of the project.

Component	Address	CIDR	Available	Host IP range
VNET	10.2.0.0	/16	65,534	10.2.0.1 - 10.2.255.254
Gateway	10.2.0.0	/27	30	10.2.0.1 - 10.2.0.30
Web	10.2.1.0	/24	254	10.2.1.1 - 10.2.1.254
SQL	10.2.2.0	/24	254	10.2.2.1 - 10.2.2.254

Table 2. Azure Network and Subnetwork IPs

In the table below the IP addresses and all relevant information has been displayed

Component	Address	CIDR	Available	Host IP range
VNET	10.3.0.0	/16	65,534	10.3.0.1 - 10.3.255.254
Gateway	10.3.0.0	/27	30	10.3.0.1 - 10.3.0.30
Web	10.3.1.0	/24	254	10.3.1.1 - 10.3.1.254
SQL	10.3.2.0	/24	254	10.3.2.1 - 10.3.2.254

Table 3. Google Cloud Network and Subnetwork IPs

A more suitable Gateway subnet CIDR range would be /29 as a /27 has a lot of wasted IP addresses. This is only the case because Azure would throw an error when a CIDR range that isn't /27 was used.

✘ To deploy a zone-redundant/zonal gateway, the GatewaySubnet must be /27 or larger.

Figure 22. CIDR range requirement for gateway

The code used for the networks and subnets will be nearly identical so only one screenshot will be provided, See figure 22.

```
resource "azurerm_virtual_network" "NETWORK-A" {
  name                = "NETWORK_A"
  address_space       = ["10.2.0.0/16"] ## keep broad as unknown
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name
}

#Create subnets for separte sections
resource "azurerm_subnet" "GatewaySubnet" { ## Gateway subnet
  name                = "GatewaySubnet" ##${random_pet(2)}
  resource_group_name = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.NETWORK-A.name
  address_prefixes     = ["10.2.0.0/27"]
}

resource "azurerm_subnet" "webVmSubnet" { ## web subnet
  name                = "webVmSubnet"
  resource_group_name = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.NETWORK-A.name
  address_prefixes     = ["10.2.1.0/24"]
}

resource "azurerm_subnet" "sqlVmSubnet" { ## sql subnet
  name                = "sqlVmSubnet"
  resource_group_name = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.NETWORK-A.name
  address_prefixes     = ["10.2.2.0/24"]
}
```

Figure 24. Azure network foundations in Terraform

To test this, I deployed the code to check the Azure Portal for the deployments as shown in Figure 23.

Name ↑↓	IPv4 ↑↓	IPv6 ↑↓	Available IPs ↑↓
GatewaySubnet	10.2.0.0/27	-	availability dependent ...
webVmSubnet	10.2.1.0/24	-	250
sqlVmSubnet	10.2.2.0/24	-	249

Figure 25. Network deployment

In order to achieve this milestone, use the commands shown in figures 26, 28, 29 and 30 on the files shown in figure 31.

```
terraform init -upgrade
```

Figure 26. Terraform Initiate

This command initiates Terraform in the local directory to enable other Terraform commands to function properly. The “-upgrade” parameter ensures that the version running is the same as the parameters set in the providers.tf file; parameters shown in figure 27.

```
terraform {  
  required_version = ">=1.2.0"
```

Figure 27. Required version parameters.

The next command in the sequence utilises Terraform plan to create a configuration plan of the main.tf file. The “-out” parameter creates an output file to save the plan to so that changes to be made can be monitored before fully deploying the solution.

```
terraform plan -out azMain.tfplan
```

Figure 28. Terraform plan

The output of this command should be similar to the message showed in the figure below.

```
Plan: 26 to add, 0 to change, 0 to destroy.  
  
Changes to Outputs:  
+ public_ip_address = (known after apply)  
+ resource_group_name = "rg"  
  
Saved the plan to: azMain.tfplan  
  
To perform exactly these actions, run the following command to apply:  
terraform apply "azMain.tfplan"
```

Figure 29. Terraform plan output

The final command for the deployment of this Terraform solution uses Terraform apply run the code configuration and attach the systems to a relative subscription. The CLI will then start sending messages corresponding to what is being deployed at the time. The configuration plan file becomes stale after deployment so another must be created to use the same command again.

```
Apply complete! Resources: 26 added, 0 changed, 0 destroyed.

Outputs:

public_ip_address = "20.224.151.215"
resource_group_name = "rg"
dylan [ ~/project-instance ]$
```

Figure 30. Terraform apply complete

Throughout the course of the project, several versions of the Terraform implementation have been taken. However, due to the two major constraints faced in this project, some of the functionality on the Google Cloud side has been stripped due to lack of mitigation for unforeseen circumstances.

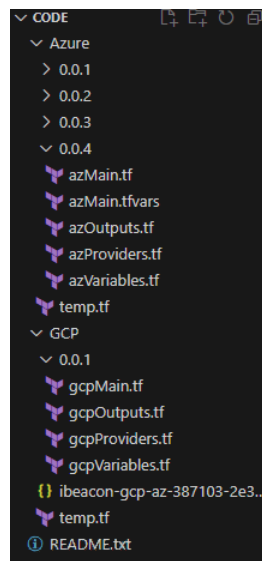


Figure 31. version control

In order to destroy the deployment, the command Terraform destroy could be used, which is more suitable for smaller projects as the plan view of the destruction can help to avoid any easily avoidable mistakes. For the destruction of resources with better practice, use the command in the following:

```
terraform plan -destroy -out azMain.tfplan
```

Figure 32. Terraform destruction plan

This command creates, or in this case overwrites the stale deployment plan left behind after the last apply and creates a destruction plan which has a similar output.

```
Plan: 0 to add, 0 to change, 26 to destroy.

Changes to Outputs:
  - public_ip_address   = "20.224.151.215" -> null
  - resource_group_name = "rg" -> null

Saved the plan to: azMain.tfplan

To perform exactly these actions, run the following command to apply:
terraform apply "azMain.tfplan"
```

Figure 33. Terraform destruction plan output

The output of this command is similar to the deployment except with different implications, see Figure 34 for the output.

```
Apply complete! Resources: 0 added, 0 changed, 26 destroyed.
```

Figure 34. Terraform deployment destroyed.

While the Network and subnets have been deployed, a virtual machine for web applications has been created on each platform to test interoperability across cloud vendors and ensure site to site connection.



<input type="checkbox"/> Name ↑↓	Type ↑↓	Subscription ↑↓	Resource group ↑↓	Location ↑↓
<input type="checkbox"/>  az-webvm1	Virtual machine	Free Trial	rg	West Europe
<input type="checkbox"/>  sql	Virtual machine	Free Trial	rg	West Europe

Figure 35. Deployed virtual machines.

As seen in Figure 35, the virtual machines have been deployed in Western Europe, as a specific zone has not been specified, these virtual machines have been deployed onto multiple regions so that geographical redundancies can be employed upon a failure of another zone in the region. Doing so successfully will ensure virtual machine uptime at an increased cost. Another form of scalability would be employing virtual machines in a scale set to match technological demands on the instance, these scale sets are deployed reactionarily for when the demand threshold gets over a certain limit more instances are deployed to share the load. (Yazdanov et al. 2013)

Due to time constraints of the project, scale sets weren't implemented.

A firewall has been implemented on both sides of the connection to monitor connections going through the vpn gateway these firewall rules consist of:

Name	Direction	Access	Protocol	Port
RDP	Inbound	Allow	Any	3389
HTTP	Inbound	Allow	Tcp	80
HTTPS	Inbound	Allow	Tcp	443
SQL	Inbound	Allow	Tcp	1433
ICMPin	Inbound	Allow	Tcp	N/A
ICMPout	Outbound	Allow	Tcp	N/A

Table 4. Basic Firewall implementation

The security rules that are important for verifying a connection through a site to site connection is RDP for remote desktop activity through a virtual machine and ICMP to enable a ping request over a vpn tunnel. Pings can also be allowed by disabling windows advanced firewall temporarily while testing although on systems that are already running this is not ideal as it's a free vulnerability for a Threat Actor to exploit.

A quota-based restraint from Google cloud platform has restricted the use of Windows virtual machines on the free trial account. While resources to combat this issue are available, the timeframe is too narrow to have made anything with the platform credits.

```
Error: Error creating instance: googleapi: Error 400: Windows VM instances are not included with the free trial. To use them, first enable billing on your account. You'll still be able to apply your free trial credits to eligible products and services., windowsVmNotAllowedInFreeTrialProject
```

Figure 36. GCP free trial restrictions.

After all of these setbacks with such little time remaining, it was decided that the Terraform solution was to have no more progress as implementing and testing was too time consuming for the little that was left.

As a result of this decision, a site-to-site vpn connection across the platforms was to be implemented manually using the respective platform's GUI to enable at least a singular form of interoperability for the scope of the project.

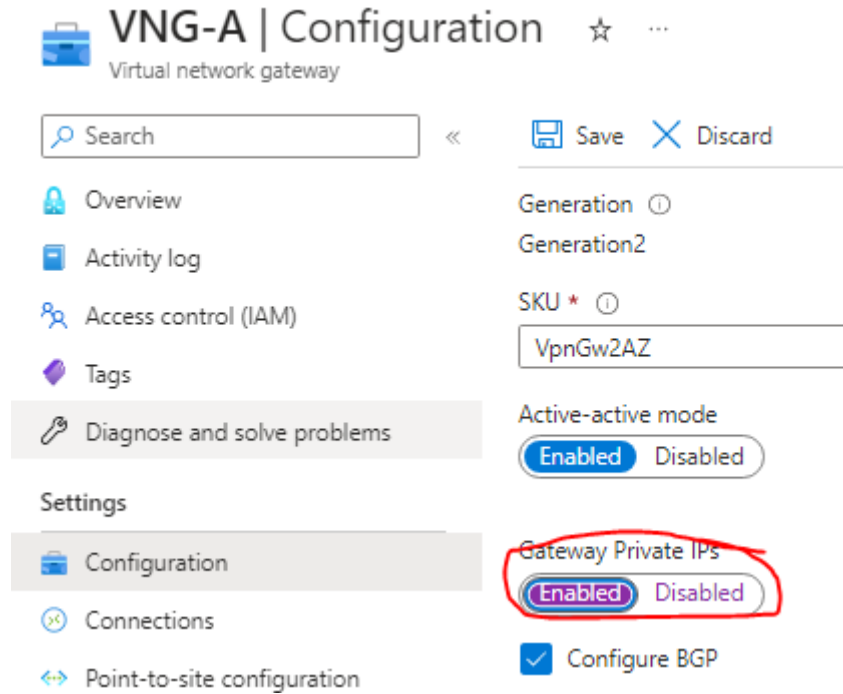


Figure 37. virtual network gateway

This virtual network gateway has been set up with the VpnGw2AZ SKU which enables a 1Gbps throughput of the gateway and allows for up to 30 unique vnet-vnet connections. It also employs the usage of an Active-Active gateways which enable high availability through cross premise connections.

The Gateway Private IPs must be enabled otherwise the site-to-site vpn gateways fail to add one another effectively through BGP peering.

BGP peering is configured with an Autonomous System Number which is a globally unique identifier that defines a group of IP prefixes used to maintain a single clearly-defined routing policy. (Battista et al. 2006)

Azure's VPN gateway has been configured with APIPA BGP Ip address set to 169.254.21.1/30 with ASN number 65002, and Google Cloud's gateway has been configured with BGP ip address 169.254.22.1/30 and ASN number 65003. The Ip addresses specified are unique because the Internet Assigned Numbers Authority has reserved the IP range of 169.254.0.0 – 169.254.255.255 which APIPA has guaranteed to have no conflict with other routable addresses. (Weil et al. 2012)

The significance of the /30 CIDR range is that there can only be 2 useable IP addresses as 2 of the 4 host address are reserved for the VPN gateway and outbound connections.

In order to get a quick solution created before the final deadline, the deployment region was changed to UK west as reducing deployment time and chances of running into errors is key for reaching the end goal at this stage.

When deploying a virtual machine of Standard_D2S_v3 to UK west, Azure had some compatibility issues with the request at the time, meaning that this virtual machine was deployed in UK south instead.

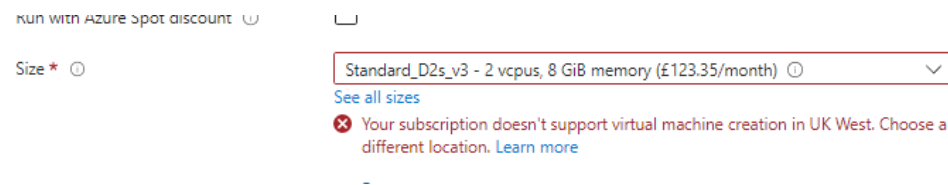


Figure 38. Virtual machine deployment error

After deploying a virtual machine of Standard_D2s_v3 to Azure and a google compute E2 to UK zones, now the site-to-site configurations can be put together and tested to ensure interoperability between vendors.

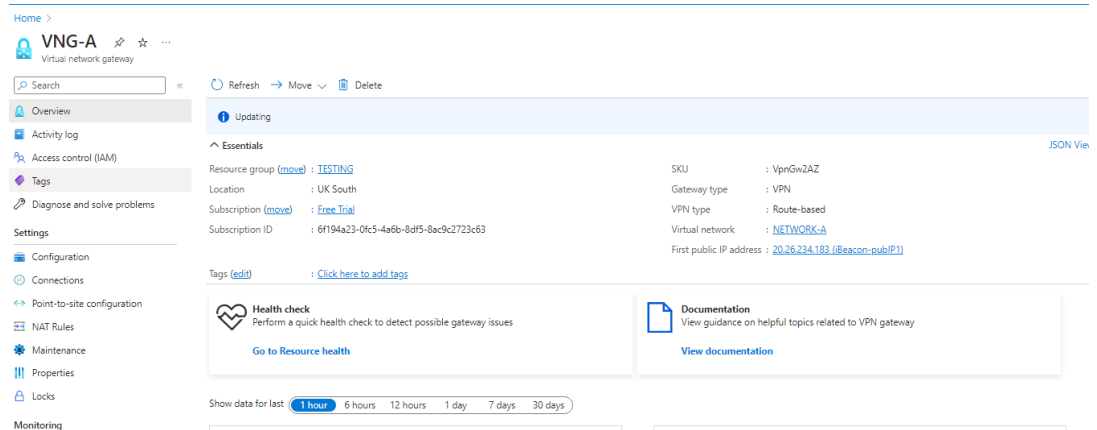


Figure 39. Vnet Appliance deployed in UK south

As seen in the figure above, the vnet gateways configurations have deployed and is ready to be tested. Before that is possible, a granular rule must be added to the firewall which allows ICMP traffic so that the built in windows ping function can test the site to site connections.

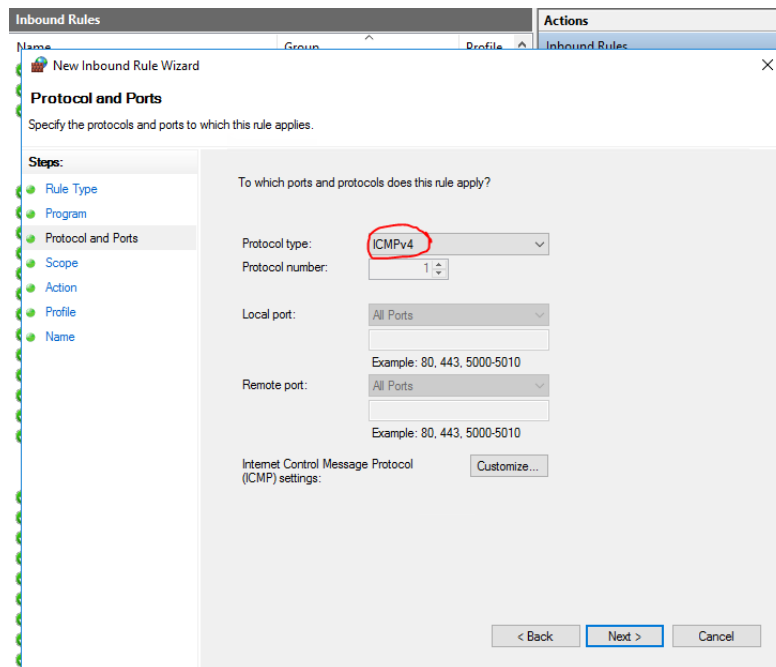


Figure 40. ICMP firewall rule.

Now that both virtual machines have been deployed, the Windows Remote Desktop Connection application can be used to take control of the virtual machine on the Azure side with which a windows authorisation prompt will pop up on the screen.

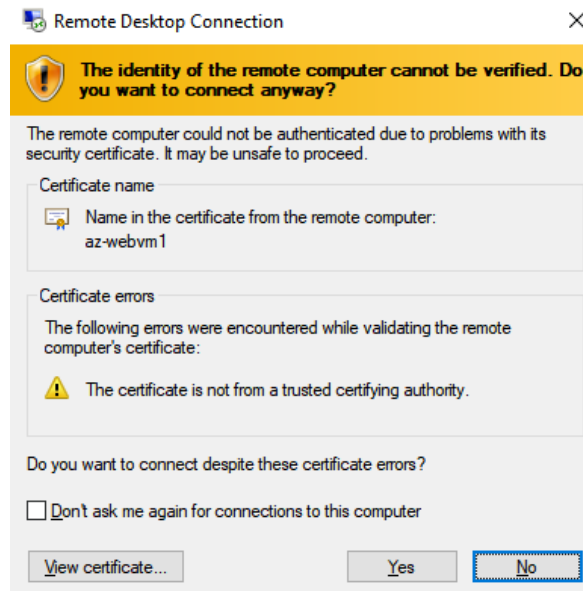


Figure 41. RDP authorisation.

After confirming this connection, it is important to bring up the command prompt in the virtual machine to check the IP configuration and ensure that its as intended.

For testing purposes only, earlier with the virtual machine deployments, an admin username and password was set for each virtual machine, the usernames being azadmin and gcadmin both with the shared password of “Testing12345678”.

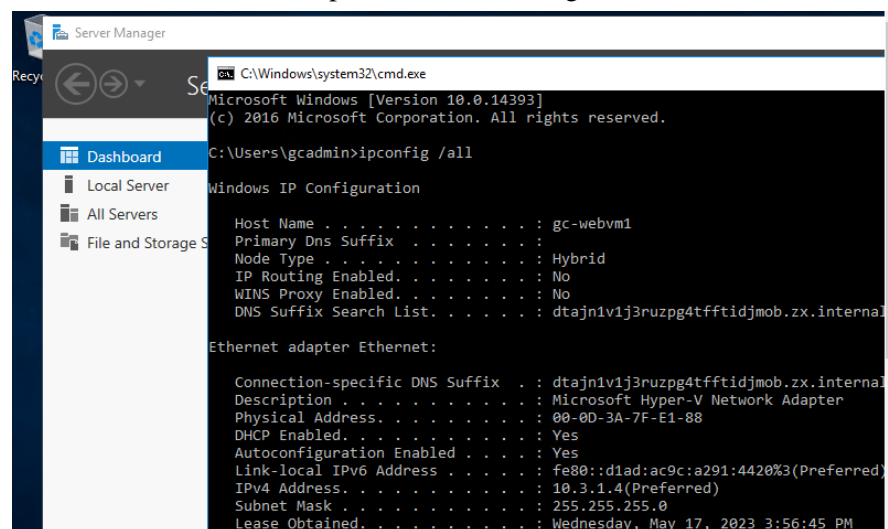


Figure 42. gc-webvm1 ipconfig from terminal

As confirmed by the gcp cmd in the figure above, the IPv4 address of 10.3.1.4 is assigned to the virtual machine as intended. The same has been done with the azure virtual machine. See figure 43.

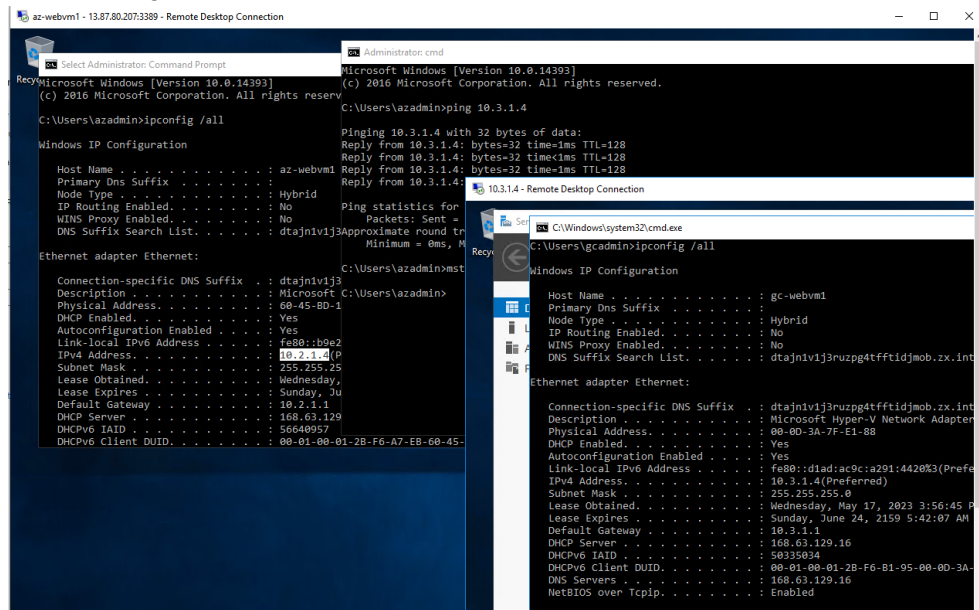


Figure 43. Working site-to-site ping.

This is the furthest this implementation has achieved in terms of functionality despite not being of full complexity as discussed at the start of the project it still mostly achieves the goal of interoperability.

Now that this connection has been tested, upon checking the Azure portal again, two vpn tunnels have shown as “CONNECTED” from the active-active tunnel configuration. Active-Active vpn connectivity acts as a cloned vpn gateway with a unique IP address which is ready to pick up a connection in the event that the other gateway goes down. This is shown in Figure 44 where the two on-premises gateways in the example are both active at the same time.

Name	Status	Connection type	Peer
VPN_TO_GCLOUD_TUN0	Connected	Site-to-site (IPsec)	GoogleCompute-PIP-1
VPN_TO_GCLOUD_TUN1	Connected	Site-to-site (IPsec)	GoogleCompute-PIP-2

Figure 44. VPN tunnels connected.

As a result of enabling BGP peering earlier, the connections table has filled up with peered IP address of each side of the site-to-site connection. See figure 45 for values.

BGP peers

Peer address	Local address	Asn	Status	Connected duration	Routes received	Messages sent
10.2.0.4	10.2.0.4	65002	Unknown	-	0	0
10.2.0.4	10.2.0.5	65002	Connected	00:50:34.1199366	0	71
10.2.0.5	10.2.0.4	65002	Connected	00:50:34.1164997	0	71
10.2.0.5	10.2.0.5	65002	Unknown	-	0	0
169.254.21.2	10.2.0.4	65003	Connected	00:21:11.4432134	3	42
169.254.22.2	10.2.0.5	65003	Connected	00:14:48.0139211	3	29
169.254.22.2	10.2.0.4	65003	Connecting	-	0	0
169.254.21.2	10.2.0.5	65003	Connecting	-	0	0

Showing only top 50 BGP learned routes in the grid, click Download learned routes above to see all.

Learned Routes

Network	Next hop	Local address	Weight	Source peer	Origin	AS path
10.2.0.0/16	-	10.2.0.4	32768	10.2.0.4	Network	-
10.3.0.0/16	169.254.21.2	10.2.0.4	32768	169.254.21.2	EBgp	65003
10.2.0.0/16	-	10.2.0.5	32768	10.2.0.5	Network	-
10.3.0.0/16	169.254.22.2	10.2.0.5	32768	169.254.22.2	EBgp	65003

Figure 45. BGP peering table.

After verifying the connection between the two sites, the network traffic monitoring system on Azure confirmed the existence of traffic through the network. In figure 46 it shows inbound and outbound traffic whilst the virtual machines were running. Its important to note that in Figure 43, inside of the azure virtual machine RDP connection, there is another virtual machine RDP connection running inside of it, hence the doubled inbound traffic in comparison to the outbound traffic.



Figure 46. Network Traffic from pinging across sites.

As a final resource collected from the implementation of this solution, a network resource visualiser has been generated to show the connectivity between cloud components on a point-to-point basis.

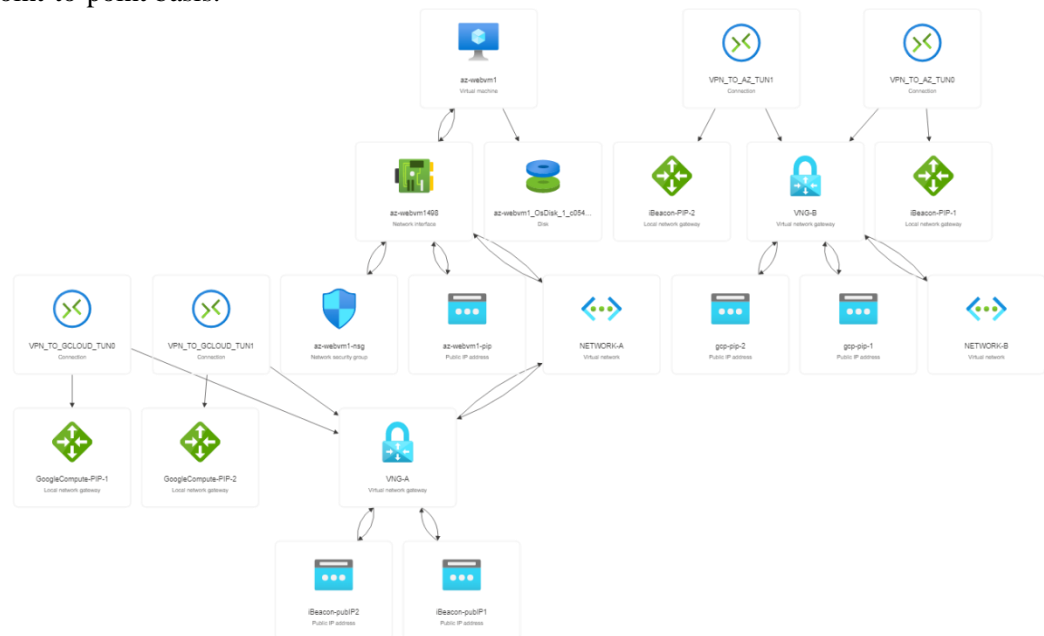


Figure 47. Azure resource visualiser.

The resource dependency would be the same on the google cloud side of the network, except just before this screenshot was taken, the google compute instance was destroyed to avoid personal billing charges.

Chapter 6

Results & Discussion

The results of the cost per feature analysis have been completed by deploying resources onto the cloud platforms and viewing the pricing statistics through the platform respective dashboards.

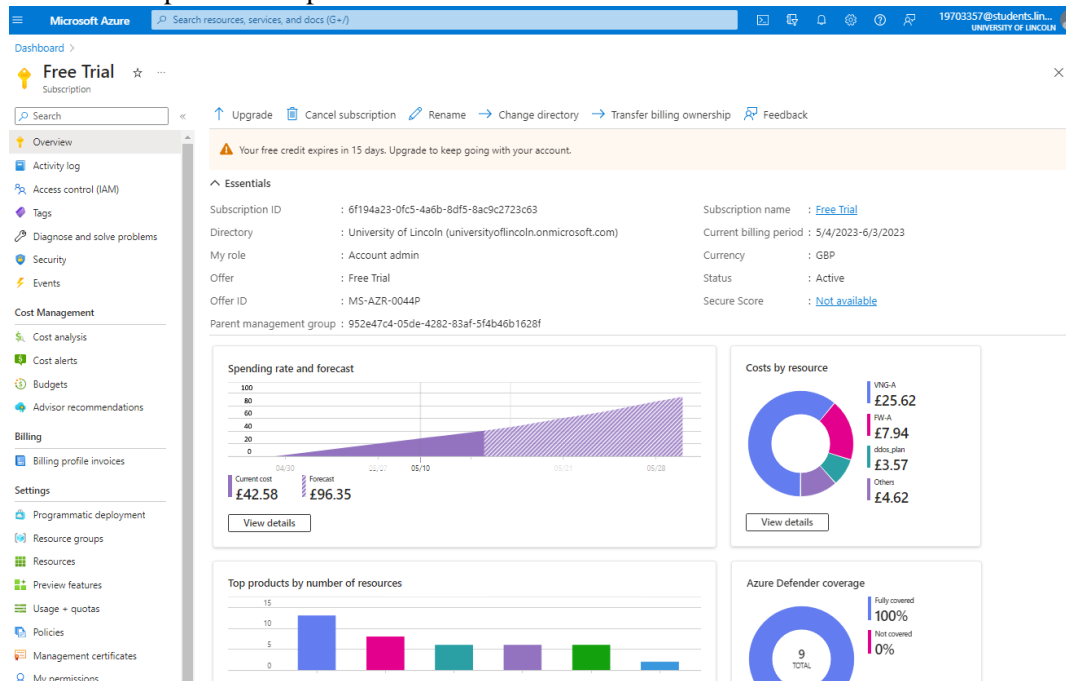


Figure 48. Subscription billing overview.

Whilst the cost per feature analysis proposed in the initial document remains quite limited as the project scope changed drastically over its life cycle, a comparison of virtual machines shall still take place.

An Azure virtual machine of Standard D2s v3 with 2 vcpus and 8 GB memory will cost £56.38 a month, whereas the google compute instance custom E2 with 2 vcpus and 2GB of memory will cost £ £50.70 a month. This is considering that 2 vcpus on a custom plan costs £33.01 a month and 8GB of memory costs £17.69 and the Azure virtual

machine is from a pay as you go plan. In terms of cheaper services for lower end specifications google cloud has the cheaper virtual machines for this exact example. However it is hard to draw conclusions to this debate when no data for networking needs has been specified. While scale sets are definitely an option to combatting this uncertainty, they are quite costly so having done some research into real world examples of network throughput would have increased the accuracy of these findings. Ultimately, the capabilities of the virtual machines and the flexibility of their plans will provide Beacon Inc with easily scalable pay as you go plans or potentially during seasons of high revenue, reserving hardware over longer periods will save money in the long run. Even considering this, Beacon Inc will need to cut overhead costs as much as possible in order to thrive in a heavily contested market.

In general this project meets the requirements set in the earlier part of this document as the correct standard for IP configurations, professional practices of setting networking resources and providing an in depth discussion on the risks and benefits of adopting cloud computing have been provided throughout the document. While some of the initial core functionality has been stripped, the main goal has been achieved through various different means as a result of unforeseen constraints and external commitments.

Chapter 7

Conclusion

Overall, the project was a success with a working site-to-site connection with relevant configurations and professional practices. Having used Terraform for the most part of an academic year, it can be said that the tool very evidently has huge benefits in the world of infrastructure as easy to learn syntax alongside the ability to reuse code to complete full deployments that follow the same configurations every time is certainly an improvement from the GUI.

While the usage of Terraform has been rewarding, the setbacks of the unforeseen circumstances such as a cancer scare and heavy commitments to other projects throughout this project have really hindered its progress

In the event that this project were to be redone, more time should be allocated to advancing knowledge on networking concepts as these helped to escalate progress in the early stages and setup higher standard foundations so that the earlier scrum sprints wouldn't have to be made redundant.

For future work, a small application for the web virtual machine and a basic database implementation could have been created to simulate a working environment. Alongside potential Azure AD policies and Role Based Access Control research aswell to add tiered access to these resources.

An even more ambitious goal would be to set up a site-to-site connection between 3 different cloud vendors.

References

- Abdollahzadegan, A., Che Hussin, A.R., Moshfegh Gohary, M. and Amini, M., 2013. The organizational critical success factors for adopting cloud computing in SMEs. *Journal of Information Systems Research and Innovation (JISRI)*, 4(1), pp.67-74.
- Al-Issa, Y., Ottom, M.A. and Tamrawi, A., 2019. eHealth cloud security challenges: a survey. *Journal of healthcare engineering*, 2019.
- Aroral, H.K., 2021. Waterfall Process Operations in the Fast-paced World: Project Management Exploratory Analysis. *International Journal of Applied Business and Management Studies*, 6(1), pp.91-99.
- Basher, M., 2019. DevOps: An explorative case study on the challenges and opportunities in implementing Infrastructure as code.
- Battista, G.D., Refice, T. and Rimondini, M., 2006, September. How to extract BGP peering information from the internet routing registry. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data* (pp. 317-322).
- Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. and Kern, J., 2001. The agile manifesto.
- Bhardwaj, S., Jain, L. and Jain, S., 2010. Cloud computing: A study of infrastructure as a service (IAAS). *International Journal of engineering and information Technology*, 2(1), pp.60-63.
- Claycomb, W.R. and Nicoll, A., 2012, July. Insider threats to cloud computing: Directions for new research challenges. In *2012 IEEE 36th annual computer software and applications conference* (pp. 387-394). IEEE.
- Chernyshev, M., Zeadally, S. and Baig, Z., 2019. Healthcare data breaches: Implications for digital forensic readiness. *Journal of medical systems*, 43, pp.1-12.
- Cunningham, W., 1992. The WyCash portfolio management system. *ACM SIGPLAN OOPS Messenger*, 4(2), pp.29-30.
- Daher, Z. and Hajjdiab, H., 2018. Cloud storage comparative analysis amazon simple storage vs. Microsoft azure blob storage. *International Journal of Machine Learning and Computing*, 8(1), pp.85-9.

- Doelitzscher, F., Fischer, C., Moskal, D., Reich, C., Knahl, M. and Clarke, N., 2012, June. Validating cloud infrastructure changes by cloud audits. In 2012 IEEE Eighth World Congress on Services (pp. 377-384). IEEE.
- Doherty, E., Carcary, M. and Conway, G., 2015. Migrating to the cloud: Examining the drivers and barriers to adoption of cloud computing by SMEs in Ireland: an exploratory study. *Journal of Small Business and enterprise development*, 22(3), pp.512-527.
- Dubey, A. and Wagle, D., 2007. Delivering software as a service. *The McKinsey Quarterly*, 6(2007), p.2007.
- GOV.UK (2022). Small to medium sized enterprise (SME) action plan. [online] GOV.UK. Available at: <https://www.gov.uk/government/publications/fcdo-small-to-medium-sized-enterprise-sme-action-plan/small-to-medium-sized-enterprise-sme-action-plan>.
- Huston, G., Rossi, M. and Armitage, G., 2010. Securing BGP—A literature survey. *IEEE Communications Surveys & Tutorials*, 13(2), pp.199-222.
- Johnson, L., Callaghan, C., Balasubramanian, M., Haq, H. and Spallek, H. (2019). Cost Comparison of an On-Premise IT Solution with a Cloud-Based Solution for Electronic Health Records in a Dental School Clinic. *Journal of Dental Education*, 83(8), pp.895–903. doi:10.21815/jde.019.089.
- Kandukuri, B.R. and Rakshit, A., 2009, September. Cloud security issues. In 2009 IEEE International Conference on Services Computing (pp. 517-520). IEEE.
- Khajeh-Hosseini, A., Greenwood, D. and Sommerville, I., 2010, July. Cloud migration: A case study of migrating an enterprise it system to iaas. In 2010 IEEE 3rd International Conference on cloud computing, pp. 450-457, IEEE.
- Larman, C., 2004. Agile and iterative development: a manager's guide. Addison-Wesley Professional.
- Lehrig, S., Eikerling, H. and Becker, S., 2015, May. Scalability, elasticity, and efficiency in cloud computing: A systematic literature review of definitions and metrics. In Proceedings of the 11th international ACM SIGSOFT conference on quality of software architectures (pp. 83-92).
- Lei, H., Ganjeizadeh, F., Jayachandran, P.K. and Ozcan, P., 2017. A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*, 43, pp.59-67.

- Li, X., Li, Y., Liu, T., Qiu, J. and Wang, F. (2009). The Method and Tool of Cost Analysis for Cloud Computing. [online] IEEE Xplore. doi:10.1109/CLOUD.2009.84.
- Mehrtak, M., SeyedAlinaghi, S., MohsseniPour, M., Noori, T., Karimi, A., Shamsabadi, A., Heydari, M., Barzegary, A., Mirzapour, P., Soleymanzadeh, M. and Vahedi, F., 2021. Security challenges and solutions using healthcare cloud computing. *Journal of medicine and life*, 14(4), p.448.
- Opara-Martins, J., Sahandi, R. and Tian, F., 2016. Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. *Journal of Cloud Computing*, 5, pp.1-18.
- Paloviita, O., 2022. Infrastructure as Code for Managed Service Providers: A Case Study.
- Petty, D.S. (2022). Exploring the Benefits of Cloud Migration and Interoperability of Cloud Vendors for Small and Medium Enterprises: Project Proposal, pp.1-2.
- Petty, D.S. (2023). Exploring the Benefits of Cloud Migration and Interoperability of Cloud Vendors for Small and Medium Enterprises: Interim Report, pp.1-4.
- Schneider, F.B., 2003. Least privilege and more [computer security]. *IEEE Security & Privacy*, 1(5), pp.55-59.
- Sinha, V., Doucet, F., Siska, C., Gupta, R., Liao, S. and Ghosh, A., 2000, September. YAML: a tool for hardware design visualization and capture. In *Proceedings 13th International Symposium on System Synthesis* (pp. 9-14). IEEE.
- Thesing, T., Feldmann, C. and Burchardt, M., 2021. Agile versus waterfall project management: decision model for selecting the appropriate approach to a project. *Procedia Computer Science*, 181, pp.746-756.
- Tsai, W.T., Sun, X. and Balasooriya, J., 2010, April. Service-oriented cloud computing architecture. In *2010 seventh international conference on information technology: new generations* (pp. 684-689). IEEE.
- Ronacher, A., 2008. Jinja2 documentation. Welcome to Jinja2—Jinja2 Documentation (2.8-dev).
- Wang, Z., Sun, K., Jing, J. and Jajodia, S., 2013, May. Verification of data redundancy in cloud storage. In *Proceedings of the 2013 international workshop on Security in cloud computing* (pp. 11-18).
- Webb, J., 2014. Network demilitarized zone (dmz).

Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C. and Azinger, M., 2012. IANA-reserved IPv4 prefix for shared address space (No. rfc6598).

Yazdanov, L. and Fetzer, C., 2013, June. Vscaler: Autonomic virtual machine scaling. In 2013 IEEE Sixth International Conference on Cloud Computing (pp. 212-219). IEEE.