KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Engineering

First cycle, 15 credits

Assessing Cloud Cost Effectiveness

A Comparative Analysis of IaaS and PaaS in Cloud Computing

GARABET ASLO
LUDWIG HAHN

Stockholm, Sweden, 2024

Assessing Cloud Cost Effectiveness

A Comparative Analysis of IaaS and PaaS in Cloud Computing

GARABET ASLO

LUDWIG HAHN

Degree Programme in Computer Engineering
Date: June 5, 2024

Supervisor: Voravit Tanyingyong
Examiner: Ki Won Sung
School of Electrical Engineering and Computer Science
Host company: Northab AB
Swedish title: Bedömning av Kostnadseffektivitet i Cloud
Swedish subtitle: En Jämförande Analys av IaaS och PaaS inom Cloud Computing

Abstract

Cloud computing has become an important part of modern Information Technology (IT) infrastructure, offering cost-effective and scalable solutions to businesses and organizations of all sizes. As businesses and organizations increasingly rely on cloud services, understanding the cost effectiveness of different deployment models becomes essential. This thesis project aims to investigate the cost effectiveness of two prominent cloud service models: Infrastructure as a Service (IaaS) and Platform as a Service (PaaS), while incorporating deployment complexities as a crucial parameter. The research conducts a comprehensive comparative analysis focusing on the cost factors associated with both IaaS and PaaS deployment models, including initial setup costs, operational expenses, scalability, resource utilization efficiency, and deployment complexities. The research methodology involves data collection, analysis, and a case study of deploying a recruiting system on different cloud platforms and running load tests to measure its performance. By assessing the cost effectiveness of IaaS and PaaS through the lens of a real-world case study, this research aims to inform decision-making processes and maximize cost efficiency while achieving business objectives. Findings reveal that IaaS offers control over infrastructure; PaaS focuses on development, minimizing management. Choice depends on on available resources, expertise, and preference for control vs. simplicity. The thesis provides valuable information on cloud computing and recommends appropriate models for different circumstances. This research expands upon current knowledge of cloud computing and deployment strategies and follows a structured approach, combining insights on cloud computing models and conducting a thorough review of existing pricing models, without introducing new solutions.

Keywords

Cloud Computing, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Deployment Complexities, Pricing Models, Scalability, Virtualization, Resource Management, Performance Optimization, Cost-effectiveness, Cloud Service Providers, Load Testing, Data Analytics.

Sammanfattning

Cloud computing har blivit en viktig del av modern IT-infrastruktur och erbjuder kostnadseffektiva och skalbara lösningar för företag och organisationer i alla storlekar. I takt med att företag och organisationer alltmer förlitar sig på molntjänster blir det avgörande att förstå kostnadseffektiviteten av de olika modeller som finns. Detta examensarbete syftar till att undersöka kostnadseffektiviteten av två framträdande molntjänstmodeller: Infrastruktur som tjänst (IaaS) och Plattform som tjänst (PaaS), samtidigt som implementeringskomplexitet inkorporeras som en avgörande parameter. Studien genomför en omfattande jämförande analys med fokus på de kostnadsfaktorer som är förknippade med både IaaS- och PaaS-modellerna. Studien tittar även på initiala installationskostnader, driftskostnader, skalbarhet, resursutnyttjande och implementeringskomplexitet. Forskningsmetodiken innefattar datainsamling, analys, en fallstudie av att implementera ett rekryteringssystem på olika molnplattformar samt köra load-tester för att mäta dess prestanda. Genom att bedöma kostnadseffektiviteten hos IaaS och PaaS genom linsen av en verklig fallstudie syftar denna studie till att stödja beslutsprocesser och maximera kostnadseffektiviteten för att nå uppsatta affärsmål. Resultaten visar att IaaS erbjuder kontroll över infrastrukturen medan PaaS fokuserar på utveckling och minimerar hanteringen. Valet beror på tillgängliga resurser, expertis och preferens för kontroll jämfört med enkelhet. Studien ger värdefull information om molnbaserad databehandling och rekommenderar lämpliga modeller för olika situationer. Denna studie breddar den nuvarande kunskapen om Cloud computing och driftsättningsstrategier och följer ett strukturerat tillvägagångssätt, kombinerar insikter om molnbaserade databehandlingsmodeller och genomför en grundlig översyn av befintliga prissättningsmodeller, utan att introducera nya lösningar.

Nyckelord

Molntjänster, Infrastruktur som tjänst (IaaS), Plattform som tjänst (PaaS), implementeringskomplexitet, prismodeller, skalbarhet, virtualisering, resurshantering, prestandaoptimering, kostnadseffektivitet, Molntjänsteleverantörer, Load-test, Data Analytics.

Acknowledgments

We would like to thank our supervisor, Voravit Tanyingyong, for his invaluable guidance and support throughout this project. Voravit's expertise and insights have been instrumental in shaping our research direction and refining our methodologies. His dedication to excellence and his willingness to provide assistance whenever needed have greatly contributed to the success of this endeavor. We are truly grateful for his mentorship and leadership.

We are also thankful for the collaboration with Northab. Their expertise and resources have significantly enriched our project, bringing new perspectives and capabilities to the table. Working alongside the Northab's team has been both productive and inspiring, and we greatly appreciate their professionalism and commitment to our shared goals.

Stockholm, June 2024
Garabet Aslo Ludwig Hahn

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
CAGR	Compound annual growth rate
CLI	Command Line Interface
CPU	Central Processing Unit
CSP	Cloud Service Provider
FinOps	Financial Operations
GCP	Google Cloud Platform
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as a Service
IP	Internet Protocol
IT	Information Technology
ML	Machine Learning
NIST	National Institute of Standards and Technology
NPM	Node Package Manager
OS	operating system
PaaS	Platform as a Service
RAM	Random-access Memory
RPS	Requests per Second
SaaS	Software as a Service
SDK	Software Development Kit
SLA	Service Level Agreement

VM	Virtual Machine
VPC	Virtual Private Cloud
VPN	virtual Private Network

Chapter 1

Introduction

In the landscape of modern computing, the rise of cloud services has revolutionized the approach businesses and organizations take toward their Information Technology (IT) infrastructure [1]. This shift involves transitioning from traditional on-premise servers to the more flexible and scalable solutions offered by cloud-based platforms. In 2022, enterprise spending on cloud infrastructure services amounted a growth of 47 billion U.S. dollars compared to the previous year. The growing market for cloud infrastructure services is driven by organizations' demand for modern networking, storage, and databases solutions [2]. Within this framework, organizations are confronted with a range of cloud computing services, each bearing significant implications for operational efficiency, scalability, and overall agility.

Among the options available, two fundamental models stand out: Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). These models offer distinct benefits and challenges, particularly in the areas of cost-effectiveness and deployment efficiency. As such, there arises a critical need for a deeper understanding of how these models perform under various operational scenarios [3].

This thesis aims to explore the comparison between IaaS and PaaS, focusing on the factors that influence their cost-efficiency and the complexities in deploying services on these platforms. By thoroughly examining and contrasting these models, the study strives to help organizations to make informed decisions about which service to use, tailored to their unique operational requirements and strategic objectives. Moreover, this thesis will examine the complexities encountered in deploying applications. This chapter will provide a brief overview of the topic, state the problem statement, clarify

the thesis's purpose, and present its goals and objectives.

1.1 Background

This thesis originates from the dynamic field of cloud computing, which has made a fundamental shift in how businesses use and manage IT resources. Northab, a development and technical recruitment company, operates within an industry with constantly evolving technological innovation and customer demands, recognizes the important role of cloud computing in driving operational efficiency, scalability, and innovation across diverse industry verticals.
Motivated by the potential of cloud technologies and the need to explore the different cloud service models, IaaS and PaaS, a collaboration has been agreed upon to find a suitable cloud service model for the deployment of a recruiting system being developed. The growing popularity of these models underscores their potential to revolutionize IT infrastructure management, offering businesses flexibility, scalability, and cost-effectiveness compared to traditional on-premises solutions. Central to Northab's interest in comparing IaaS and PaaS is the recognition of the distinct advantages and trade-offs associated with each model. While IaaS offers businesses the flexibility to provision and manage virtualized infrastructure components such as servers, storage, and networking resources [4], PaaS abstracts away the underlying infrastructure complexities, enabling developers to focus on building and deploying applications rapidly [5].
By conducting a comprehensive comparison between IaaS and PaaS, Northab aims to attain insights into the nuanced differences between these two models, understanding their respective strengths, weaknesses, and use cases. Through this thesis, necessary knowledge and insights will be provided to navigate the complexities of cloud service selection, facilitate the optimization of IT infrastructure and fostering sustainable growth in an increasingly technology-driven business environment.

1.2 Problem

Cloud computing offers the promise of eliminating the need for costly on-premise computing facilities. Through virtualization and resource time-sharing, clouds efficiently serve a diverse user base with varying needs using a single set of physical resources. High performance is a crucial

advantage of cloud computing, essential for both users and service providers. However, evaluating cloud services has become increasingly challenging with the proliferation of cloud providers [6]. This complexity arises from the unique characteristics, types, and models of cloud environments. The choice between IaaS and PaaS has emerged as a critical decision point. This decision impacts not only the technical architecture but also the operational flexibility, scalability, and cost-efficiency of IT infrastructure. While both models offer distinct advantages, the lack of clear, comparative insights into their cost-effectiveness and deployment complexities poses a significant challenge for businesses aiming to optimize their cloud strategy. **How can organizations effectively navigate the choice between IaaS and PaaS to optimize their cloud computing strategy in terms of cost-efficiency and deployment complexity?**

1.3 Purpose

The purpose of this thesis is to compare IaaS and PaaS in the context of Northab's operational needs. Simultaneously, the study contributes to the academic and engineering discourse by providing practical insights into the application of cloud services in business environments.

1.4 Goals

The goal of this thesis project is to investigate the cost effectiveness IaaS and PaaS cloud service models, with a focus on deployment complexities as a parameter. This objective has been broken down into the following three sub-goals:

1. Conduct a comprehensive comparative analysis of the cost factors associated with IaaS and PaaS deployment models. This analysis will encompass initial setup costs, operational expenses, scalability, resource utilization efficiency, and deployment complexities.

2. Evaluate the case study and the deployment of the recruiting system. This real-world deployment scenario will serve as a practical lens through which to assess the cost effectiveness of IaaS and PaaS solutions.

3. Provide valuable insights and recommendations to inform decision-making processes regarding the selection of appropriate cloud comput-

ing models for different circumstances. By maximizing cost efficiency while achieving business objectives.

1.5 Research Methodology

In this study, different methodologies are employed to comprehensively investigate the cost effectiveness of cloud service models and deployment strategies.

1. **Review of cloud computing models:** This methodology involves conducting a thorough review of different cloud models, including IaaS and PaaS. This review incorporates an overview of the services and resources currently available within each model. By examining existing literature, industry reports, and expert opinions, this methodology aims to establish a foundational understanding of the key concepts and components of cloud computing.

2. **Case study of Northab's recruiting system:** A case study approach focusing on Northab's recruiting system. This involves deploying the system to different cloud service providers to test the various cloud services and collect relevant data. By using Northab's system as a real-world example, this methodology allows for practical insights into the cost effectiveness and performance of different cloud services in a specific context. Through data collection during the deployment process, This approach offers valuable real-world data to guide the analysis.

3. **Data analysis:** This methodology involves analyzing the data collected from the review of cloud models, as well as the tests conducted on different cloud services during the deployment of Northab's Recruiting System. This analysis involves integrating the insights gained from the review with the data collected from the case study. By examining factors such as initial setup costs, operational expenses, scalability, and deployment complexities, this methodology aims to evaluate the cost effectiveness of IaaS and PaaS models. Statistical and descriptive analysis will be employed to analyze the data and draw meaningful conclusions.

By employing a combination of these methodologies, this study seeks to provide a comprehensive and nuanced analysis of the cost effectiveness of

cloud service models. Each methodology contributes unique insights, with the review providing theoretical foundations, the case study offering practical insights, and the data analysis facilitating evidence-based conclusions. This multi-method approach enhances the validity and reliability of the findings, ultimately informing decision-making processes related to cloud deployment strategies.

1.6 Delimitations

As this study explores cloud computing cost effectiveness within the context of IaaS and PaaS models, it is essential to outline the boundaries within which this study operates.

1. One such boundary entails the specificity of cloud service providers examined in this study. While the cloud computing landscape boasts a numerous of providers, our analysis will focus on a select few, recognizing the impracticality of testing all available options. Consequently, our study will not encompass a comprehensive evaluation of all cloud providers but will instead concentrate on a subset considered representative of the broader environment.

2. The study will primarily focus on cloud computing cost-effectiveness within the area of Europe. It will not encompass a global analysis of cloud service providers and their cost-effectiveness in various regions.

3. The analysis will be conducted within a specific time frame, considering the current market conditions and pricing structures of the selected cloud service providers. Historical data or future projections beyond this time frame will not be included in the study.

1.7 Benefits, Ethics, and Sustainability

In addition to technical and operational considerations, it is important to address the broader implications of cloud computing in terms of benefits, ethics, and sustainability. Cloud computing offers many benefits beyond cost savings. It enables businesses to scale resources dynamically, reducing the need for upfront capital investment and improving agility. Furthermore, cloud services often come bundled with advanced security features, providing enhanced data protection compared to traditional on-premises solutions. From

a business perspective, cloud adoption can facilitate innovation by allowing rapid experimentation and deployment of new services, fostering a competitive edge in the market.
As businesses leverage cloud computing to store and process vast amounts of data, ethical considerations regarding data privacy, security, and ownership become paramount. Companies must adhere to stringent data protection regulations and ethical guidelines to ensure the privacy and rights of individuals are respected. Additionally, there are ethical concerns surrounding the environmental impact of cloud infrastructure, particularly concerning energy consumption and carbon emissions [7]. Addressing these ethical considerations requires transparent policies, robust security measures, and responsible resource management practices.
The sustainability of cloud computing infrastructure is increasingly scrutinized as the digital economy continues to expand. While cloud services offer potential energy and resource efficiencies compared to on-premises data centers, the rapid growth of cloud infrastructure poses environmental challenges. Data centers consume significant amounts of energy, primarily from non-renewable sources, contributing to carbon emissions and environmental degradation. Promoting sustainability in cloud computing involves investing in renewable energy sources, optimizing data center efficiency, and adopting eco-friendly practices throughout the cloud lifecycle.

1.8 Structure of the thesis

The structure of the thesis encompasses several chapters, each serving a distinct purpose in presenting the research comprehensively. Chapter 2, delves into relevant background information, including major areas of focus, subareas, and related work. This thorough exploration helps establish a foundation of understanding for readers.
Methodology is addressed in Chapter 3, where the research process, paradigm, data collection methods, experimental tests, and planned data analysis techniques are outlined. This chapter explains the approach taken to address the research problem and ensures transparency in the research process.
Chapter 4, Implementation, details the practical aspects of the research, including software design, coding examples, and simulation models. This section bridges the gap between theory and practice, providing insights into the application of the research findings.
Results and analysis are presented in Chapter 5, where the major findings of the research are discussed and interpreted. This chapter offers insights into the

significance of the results and their implications for the field of study.
Finally, the conclusions and future work in Chapter 6 summarizes the key conclusions drawn from the research and outlines potential avenues for future investigation. This chapter reflects on the limitations of the study and offers recommendations for further research, ensuring the research contributes meaningfully to the field.

Chapter 2

Background

Cloud computing has rapidly evolved, similar to the transformation brought about by personal computers and the internet. Initially adopted by startups and small businesses, it has now gained widespread acceptance, with enterprise budgets for cloud initiatives growing substantially. Success stories like Netflix and Instagram highlight the increasing adoption of cloud solutions [8].
Before cloud computing, organizations relied on maintaining their own computer systems and physical infrastructure housed within their premises, typically referred to as data centers[9]. These data centers contained all the necessary equipment, including servers, storage units, switches, networks, routers, and more. Organizations had to buy and manage their own hardware and software infrastructure, which involved significant upfront costs and ongoing maintenance expenses, less flexibility and higher risk of vulnerabilities to physical security threats.
Cloud computing is a distributed computing paradigm that focuses on providing a wide range of users with distributed access to scalable, virtualized hardware and/or software infrastructure over the internet [10]. Cloud computing offers organizations scalability, allowing them to adjust resources based on demand for enhanced flexibility and cost efficiency, particularly beneficial for businesses with fluctuating workloads or rapid growth. Additionally, its cost-effectiveness is evident through the elimination of upfront investments, shifting to a pay-as-you-go model that fosters cost savings and financial predictability. Moreover, cloud computing improves accessibility and collaboration by centralizing data and applications, enabling remote access and seamless collaboration across distributed teams.

2.1 Introduction to Cloud Computing

Cloud computing offers on-demand scalable, reliable, and device-independent services to its users. Clouds represent distributed technology platforms leveraging advanced innovations to provide highly resilient and scalable environments, remotely accessible by organizations for various purposes [11]. By facilitating resource access via web-based tools and applications over the internet, cloud computing revolutionizes the delivery of IT services, eliminating the need for direct server connections. Unlike traditional storage methods, cloud-based storage allows files to be securely saved to remote databases via the internet, rather than on proprietary hard drives or local storage devices.
National Institute of Standards and Technology (NIST) characterizes cloud computing as agile, scalabile, cost-efficient, and accessible [12]. Building upon NIST's framework, the key characteristics that define cloud computing are:

- **On-demand self-service:** Cloud computing enables users to provision computing resources such as storage, processing power, and applications without requiring human intervention from service providers. Users can dynamically adjust resource allocation based on their immediate needs, enhancing flexibility and scalability.

- **Broad network access:** Cloud services are accessible over the internet from a wide range of devices including desktop computers, laptops, tablets, and smartphones. This accessibility facilitates remote access to cloud resources, enabling users to work from anywhere with an internet connection.

- **Resource pooling:** Cloud providers aggregate and optimize computing resources to serve multiple users simultaneously. This pooling of resources allows for efficient resource utilization and optimization, reducing costs and improving overall performance.

- **Measured service:** Cloud providers services are typically metered, meaning that users only pay for the resources they consume. Usage is measured and billed based on factors such as storage capacity, processing power, bandwidth, and active user accounts. This pay-per-use model offers cost transparency and allows users to scale resources according to their budget and requirements.

- **Rapid elasticity:** Capabilities can be provisioned and released elastically, sometimes automatically, to rapidly scale up or down computing resources such as processing power and storage capacity in response to changes in demand, ensuring optimal performance and cost efficiency.
- **Device independence:** Accessabilty from a variety of devices with internet connectivity, regardless of the operating system (OS) or hardware specifications. This device independence allows users to access cloud resources from any device, enhancing flexibility, and mobility in accessing applications and data.

Cloud computing offers a range of service models tailored to meet diverse business requirements and IT needs. Fundamental concepts of IaaS, PaaS, and Software as a Service (SaaS) are furthur explained below. These service models provide organizations with unparalleled flexibility, scalability, and cost-effectiveness, catering to diverse business needs and objectives. By understanding the distinct characteristics of each service model, organizations can strategically harness the power of cloud computing to drive innovation, streamline operations, and achieve their business goals.

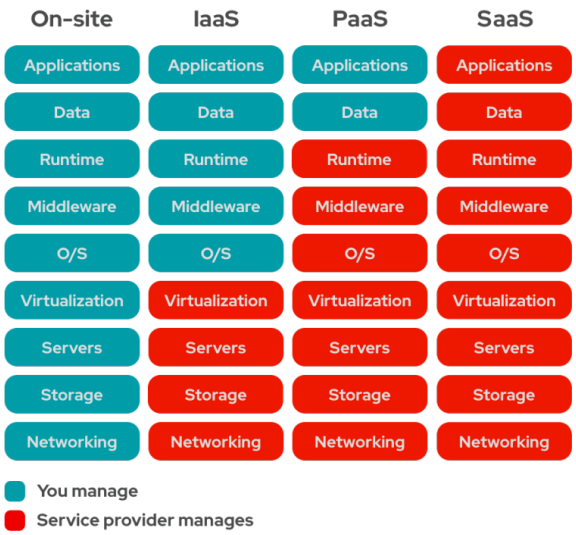

Figure 2.1: Comparison of Management Responsibilities in IaaS, PaaS, SaaS and On-Site Deployment Models [13].

2.1.1 Infrastructure as a Service (IaaS)

IaaS provides virtualized computing resources over the internet, including virtual machines, storage, and networking. Revenue of IaaS is expected to show an annual growth rate (Compound annual growth rate (CAGR) 2024-2028) of 16.52% [14]. With IaaS, organizations can provision and manage infrastructure components on-demand, eliminating the need for physical hardware investments. Users have full control over the OSes, applications, and development frameworks, enabling greater flexibility and scalability. IaaS is ideal for businesses seeking to offload infrastructure management tasks and achieve cost savings through pay-as-you-go pricing models. Most popular providers of IaaS are Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP).

2.1.2 Platform as a Service (PaaS)

PaaS offers a complete development and deployment environment in the cloud, empowering developers to build, deploy, and manage applications without the complexity of infrastructure management [12]. PaaS providers like Heroku, Microsoft Azure App Service and Google App engine, offer a suite of tools, frameworks, and middleware services to streamline the development lifecycle, from coding to testing and deployment. By abstracting away infrastructure concerns, PaaS accelerates application development, enhances collaboration among development teams, and enables rapid iteration and innovation. PaaS is well-suited for organizations focused on software development and seeking to expedite time-to-market for new applications.

2.1.3 Software as a Service (SaaS)

SaaS delivers software applications over the internet on a subscription basis, eliminating the need for users to install, manage, and maintain the applications locally [12]. Users access SaaS applications through web browsers or APIs, enjoying seamless updates, maintenance, and support from the provider. SaaS offerings span a wide range of applications, including productivity tools, customer relationship management (CRM) software, and enterprise resource planning (ERP) systems. Some examples of popular SaaS applications include, SalesForce, Zoom and Dropbox [15]. SaaS provides organizations with immediate access to powerful software solutions, scalability to accommodate user growth, and simplified management and licensing. SaaS won't be considered in this study because the research

focuses on the cost-efferctiveness and complexities associated with deploying applications to the cloud. SaaS offerings are software applications delivered over the cloud to end-users, rather than platforms or infrastructure targeted at developers for application deployment.

2.2 Future of Cloud Computing

Cloud computing leads the way for technological innovation, ready to reshape the way businesses operate and thrive in the digital age. As we look ahead, the future of cloud computing promises to be characterized by higher levels of scalability, agility, and efficiency, driven by emerging trends and transformative technologies [16].

One of the key drivers of this evolution is the growth of edge computing. The growth of connected devices and IoT sensors, brings computing closer to data sources [17]. This real-time processing improves decision-making and application responsiveness. As edge computing grows, it merges with cloud technologies, forming a distributed computing model spanning centralized clouds and edge devices.

cloud computing will also see widespread adoption of cloud-native technologies and open-source solutions. Platforms like Kubernetes and Istio [18] have already transformed container management and microservices architecture, enabling organizations to deploy applications with high speed [17]. Data analytics and artificial intelligence will also play an important role. With the rise of data volumes, cloud-based analytics tools are helping organizations extract actionable insights. Machine learning algorithms, powered by the cloud's scalability, are automating tasks and driving innovation.

Furthermore, cloud computing will emphasize standardization, interoperability, and openness. Cloud platforms are becoming the foundation of enterprise IT, leading to systemic changes in technology deployment. Standardized platforms enable automation and efficiency, while interoperability ensures seamless integration between various cloud environments and legacy systems.

2.3 Cloud Computing Pricing Models

One of the key considerations for organizations adopting cloud services is understanding the pricing models offered by cloud providers. Cloud pricing models dictate how users are charged for utilizing cloud resources and services, playing a crucial role in optimizing costs and aligning expenses

with business needs. Different cloud service providers adopt various pricing models. However, the most common model in cloud computing is the ”pay-as-you-go” model. Under this model, customers pay a fixed price for each unit of usage. For instance, Amazon [19], a prominent player in the cloud computing market, employs this model by charging a fixed rate for every hour of virtual machine usage. Similarly, other industry leaders such as Microsoft Azure [20], also embrace the ”pay-as-you-go” approach.

2.3.1 Pay-as-you-go Model

The Pay-as-you-go pricing model is based on the real-time consumption of cloud resources, encompassing compute resources, storage space, network bandwidth, and data transfer [21]. Users pay for the resources utilized, typically on an hourly or per-unit basis, offering flexibility and cost control. While this model ensures transparency by providing customers with the exact price to be paid and reserves resources for their designated period, it may pose challenges. There's a risk of overpayment, as users may pay for more cloud resources (processing power, storage, etc.) than they actually need, leading to wasted money. Moreover, the static pricing model lacks flexibility in adjusting prices based on fluctuating demand, potentially resulting in users paying higher prices during low-demand periods.

2.3.2 Subscription Model

In the subscription-based pricing model, users subscribe to a predetermined service plan or tier, paying a fixed monthly or annual fee for access to a specific level of cloud resources and services [21]. Pricing is based on the duration of the subscription, offering a static cost structure. While subscription plans often provide discounted rates compared to pay-per-use pricing, they may present challenges for users. Customers may sometimes overpay or underpay for the resources reserved, depending on their actual usage. If users extensively utilize the reserved resources, they might underpay, whereas if they under utilize the resources, they may end up overpaying. Despite these considerations, subscription-based pricing is suitable for organizations with predictable workloads and budgetary requirements.

2.3.3 Reserved Instance Model

At its essence, the Reserved Instance pricing model centers on reserving instances. Customers commit to a set amount of compute capacity for

a defined period, typically one to three years [21], unlocking substantial discounts compared to on-demand pricing. This proactive stance enables businesses to better align cloud spending with operational needs and budget limitations. Notably, it offers significant cost savings for long-term commitments, making it well-suited for stable, predictable workloads. However, drawbacks include limited flexibility and the risk of resource waste if business requirements shift.

2.4 Deployment complexities in IaaS and PaaS

When considering deployment complexities in IaaS and PaaS models, organizations encounter distinct challenges and advantages unique to each model.

2.4.1 IaaS Deployment complexities

In IaaS, organizations are responsible for managing the deployment of virtualized infrastructure components, such as virtual machines, storage, and networking resources. While cloud providers handle the underlying physical infrastructure, users must still configure, deploy, and manage their virtualized environments.

Deployment complexities in IaaS typically involve tasks such as selecting appropriate instance types, configuring networking settings, setting up storage volumes, and installing OSes and applications. Users have granular control over the configuration and customization of their virtual environments, allowing for flexibility and customization according to specific requirements. However, the self-service nature of IaaS also means that organizations bear the responsibility of ensuring security, compliance, and performance optimization. This includes tasks such as implementing security measures, monitoring resource usage, and optimizing infrastructure for cost-effectiveness.

Overall, while IaaS offers flexibility and control over infrastructure deployment, it requires significant expertise and effort from organizations to effectively manage and optimize their virtualized environments.

2.4.2 PaaS Deployment complexities

In contrast, PaaS abstracts away the underlying infrastructure, allowing users to focus solely on deploying and managing their applications and services. PaaS providers handle tasks such as provisioning and scaling of infrastructure components, allowing users to concentrate on application development and deployment.
Deployment complexities in PaaS primarily revolve around configuring and deploying applications within the platform environment. Users typically interact with platform-specific tools and interfaces to upload application code, configure runtime environments, define service dependencies, and manage application lifecycle processes. PaaS platforms often offer built-in features for automated scaling, load balancing, and application monitoring, simplifying deployment and management tasks. This streamlines the deployment process and accelerates time-to-market for applications, as users can leverage pre-configured infrastructure and services.
However, adopting a PaaS solution may tie organizations to a specific Cloud Service Provider (CSP)'s ecosystem, limiting their flexibility to switch providers in the future. Another downside of PaaS is limited control over infrastructure, which may be problematic if there are specific performance, security, or compliance requirements that cannot be met within the borders of the PaaS platform.

2.5 Summary

Cloud computing has undergone rapid evolution and gained widespread acceptance, with enterprise budgets for cloud initiatives growing substantially. Before cloud computing, organizations relied on maintaining their own on-premises data centers, managing all necessary equipment, including servers, storage units, and networking components. However, cloud computing has revolutionized IT infrastructure by providing on-demand, scalable, and device-independent services over the internet. It offers organizations scalability and cost-efficiency, eliminating the need for upfront investments. Additionally, cloud computing improves accessibility and collaboration by centralizing data and applications, enabling remote access and seamless collaboration across distributed teams. Cloud computing offers a range of service models, including IaaS, PaaS and SaaS, each tailored to meet diverse business requirements and IT needs. Fundamental concepts of these service models provide organizations with unparalleled flexibility, scalability, and

cost-effectiveness.

Cloud computing pricing models play a crucial role in optimizing costs and aligning expenses with business needs. The most popular models for IaaS include Pay-as-you-go and Hybrid pricing, while Subscription model is predominant for PaaS. IaaS Pay-as-you-go pricing offers flexibility but may lead to overpayment, while the Hybrid model balances costs and scalability. PaaS Subscription pricing offers a static cost structure but may result in overpayment or underpayment depending on resource usage. Deployment complexities differ between IaaS and PaaS, with IaaS requiring significant expertise for managing virtualized environments and PaaS simplifying deployment tasks through abstracted infrastructure. However, IaaS is more flexible and suitable for applications with specific or unique requirements that may not be fully supported by PaaS offerings.

By understanding the distinct characteristics of the cloud service and pricing models, organizations can strategically harness the power of cloud computing to drive innovation, streamline operations, and achieve their business goals.

Chapter 3

Method

This chapter serves as a comprehensive guide to the research methodology employed in this thesis. It begins with an outline of the research process in Section 3.1, detailing the steps undertaken to conduct the study effectively. Following this, Section 3.2 delves into the research paradigm adopted, providing insight into the underlying theoretical framework guiding the research approach.
Section 3.3 focuses on the data collection techniques used, elaborating on the sources from which data was gathered and the parameters considered. Specifically, Section 3.3.1 discusses the various data sources used, while Section 3.3.2 explores the key data parameters analyzed throughout the study.
In Section 3.4, the experimental design employed in the research is elucidated, with particular attention given to the Locust tool and the metrics utilized for assessment, as detailed in Sections 3.4.1 and 3.4.2, respectively.
Assessment of the reliability and validity of the collected data is expounded upon in Section 5.7, with discussions on the methods employed to ensure data accuracy and consistency. Validity and reliability are discussed separately in Sections 5.7.1 and 5.7.2, respectively.
Moving forward, Section 3.5 outlines the planned data analysis methodology, encompassing both statistical and descriptive data analysis techniques. Specifically, Section 3.5.1 delves into statistical data analysis, while Section 3.5.2 elaborates on descriptive data analysis methods.
Finally, the chapter concludes with an exploration of the evaluation framework chosen to assess the research findings, as detailed in Section 3.6. This framework provides a structured approach to evaluate the key aspects of the research outcomes, ensuring a comprehensive and robust analysis.

3.1 Research Process

This study employs both theoretical and practical approaches in its research process. It begins with a thorough review of existing literature and theoretical frameworks to establish foundational knowledge. Subsequently, practical aspects are explored through a case study and tests to validate theoretical concepts and gather important data.
The activity diagram in the Figure 3.1 illustrates the research and development process involved in investigating the cost effectiveness of cloud service models and deployment strategies. It illustrates the steps taken from academic interest to addressing the original problem.

1. **Review of different cloud models:** In this step, a comprehensive review of different cloud models is conducted. This involves examining existing literature, industry reports, and expert opinions to establish foundational knowledge.

2. **Case Study and Tests:** A case study approach focusing on Northab's recruiting system is undertaken. This involves deploying the system to different cloud service providers to test various cloud services and collect relevant data.

3. **Data Collection:** Practical insights into cost effectiveness and performance are gained through observations and data collection.

4. **Data Analysis:** The data collected from the case study and tests is analyzed. Factors such as initial setup costs, operational expenses, scalability, and deployment complexities are examined to evaluate the cost effectiveness of IaaS and PaaS models.

5. **Integrating Insights:** This step involves integrating insights obtained from the review of cloud models with the data collected and analysed during the case study of Northab's Recruiting System. By combining theoretical knowledge with practical findings, a comprehensive review of cost effectiveness is achieved.

6. **Conclusions and Recommendations:** Based on the integrated insights, conclusions are drawn regarding the cost effectiveness of different cloud service models and deployment strategies. Recommendations for optimal cloud utilization will also be provided.

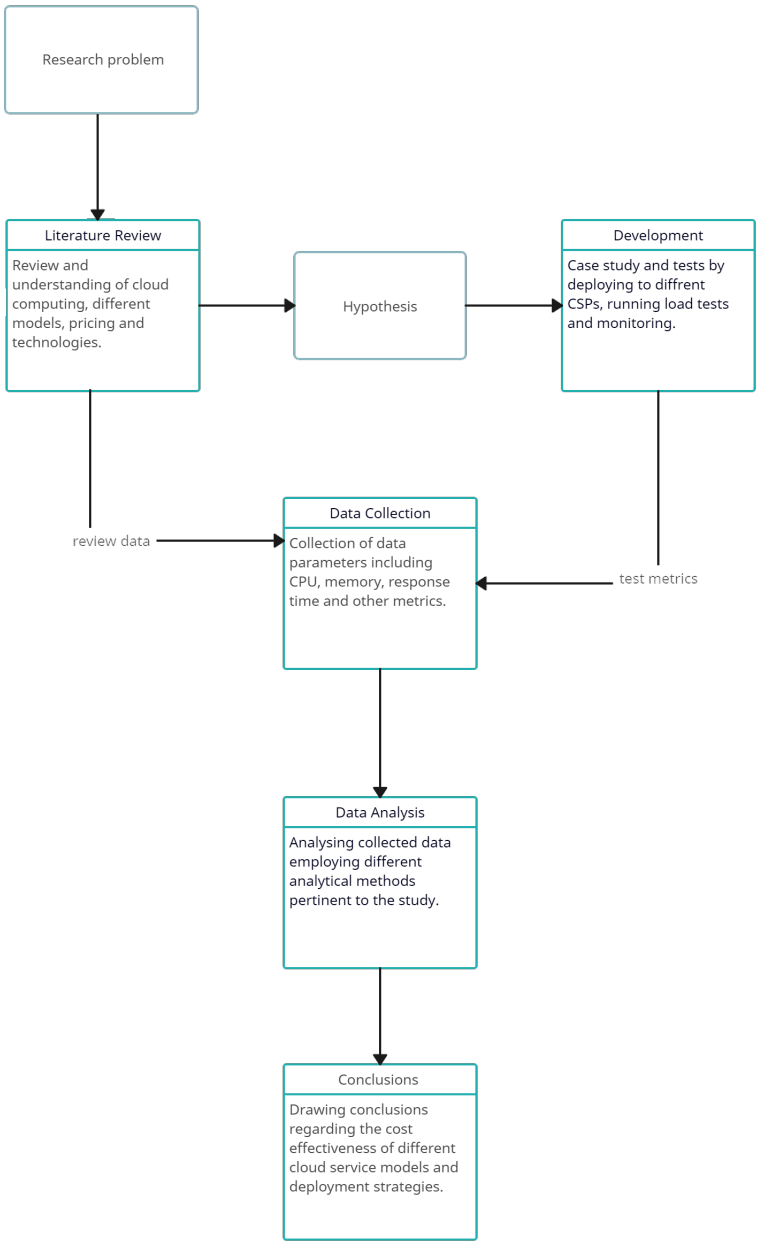

Figure 3.1: Research and Development Process Activity Diagram

3.2 Research Paradigm

In this study, various research designs and paradigms are used, including both qualitative and quantitative methodologies. These diverse approaches enable a comprehensive exploration of the research questions from different perspectives and facilitate a more nuanced review of the investigation. By integrating qualitative methods, such as reviews and literature studies, with quantitative techniques like case studies and data analysis, the study captures both the depth and breadth of the research topic. This multi-method approach enhances the validity and reliability of the findings, offering a more robust foundation for discussions, drawing conclusions and making recommendations.

3.3 Data Collection

In order to compare pricing and deployment complexities of cloud computing using IaaS and PaaS, a case study approach focusing on Northab's recruiting system will be undertaken. This involves deploying the system to different CSPs to test various cloud services and collect relevant data. The implementation of robust data collection procedures is essential for this aim. This section outlines the methodologies employed to gather relevant information regarding the deployment, usage, and associated costs of applications across various CSPs. By thoroughly reviewing pricing documentation, observing deployment complexities, and tracking costs, the collected data will serve as the foundation for the analysis.

3.3.1 Data Sources

Primary data is derived directly from the official documentation and websites of the cloud service providers being assessed. This involves a thorough examination of technical specifications, Requests per Seconds (RPSs), pricing models and feature sets. In some cases, these websites provide pricing calculators to estimate the cost of the architecture solution [22], but in this research a more accurate cost estimation will be done by deploying and running tests on the recruiting system of the case study. By accessing official documentation, the study ensures the accuracy and authenticity of the information obtained, capturing the latest developments in services, features and updates.

Performance-related data will be collected by conducting front-end load time benchmarks using Locust [23], an open source load testing tool on a standardized machine. This method is designed to assess the Central Processing Unit (CPU) and Random-access Memory (RAM) usage required for the front-end of the recruiting system to function under varying loads. The CPU and RAM resources utilized by a deployed application are key factors influencing its chosen model and pricing structure. This testing procedure will apply uniform conditions across all CSPs, ensuring consistent and dependable results across diverse environments.
Secondary data is obtained from reliable industry reports, publications, and research papers that focus on cloud computing and the individual providers being analysed. These sources offer valuable insights into various aspects, including market trends, competitive landscapes, customer satisfaction ratings, and performance benchmarks. By leveraging such industry reports, the analysis gains depth and breadth, incorporating expert opinions and market perspectives.

3.3.2 Data Parameters

In analyzing cloud service providers, several key data parameters are used for comprehensive evaluation. These parameters encompass aspects related to capacity, availability/performance, serviceability, and cost.
Capacity parameters encompass the computational resources available to users, including CPU, memory, and disk space. CPU capacity determines the processing power available for running applications and handling workloads. Memory capacity, also known as RAM, influences the system's ability to efficiently manage and process data. Disk capacity refers to the storage space available for storing data and applications. Assessing the capacity parameters involves evaluating the scalability, flexibility, and performance of the resources provided by cloud service providers.
Availability and performance parameters focus on the reliability and responsiveness of cloud services during runtime. Availability measures the uptime and accessibility of services, ensuring uninterrupted operation for users. Performance metrics, such as response time and throughput, evaluate the speed and efficiency of data processing and application execution. Analyzing availability and performance parameters involves examining Service Level Agreements (SLAs) [24], uptime guarantees, and performance benchmarks to assess the reliability and responsiveness of cloud platforms.
Serviceability encompasses the ease of managing, maintaining, and trou-

bleshooting cloud services. It includes features such as monitoring, logging, automation, and support services provided by cloud service providers. Serviceability parameters evaluate the tools, interfaces, and support mechanisms available for managing and troubleshooting cloud environments. Assessing serviceability involves examining documentation, support options, and user feedback to determine the level of assistance and resources available for managing and maintaining cloud services.
Cost parameters focus on the financial implications of using cloud services, including Virtual Machine (VM) costs, data transfer costs, and storage costs. VM costs are associated with the provisioning and usage of virtual computing instances, typically charged on a per-hour or per-month basis. Data transfer costs involve charges incurred for transferring data between cloud services, regions, or networks. Storage costs relate to the fees for storing data in cloud storage services, often based on the amount of data stored and the storage tier selected. Analyzing cost parameters involves comparing pricing models, calculating total cost of ownership (TCO), and considering factors such as discounts, usage patterns, and cost optimization strategies to determine the most cost-effective solution.

3.4 Experimental design

3.4.1 Locust

Locust was selected as the benchmarking tool due to its robustness and open-source nature. Developed with a focus on load testing, Locust allows for the simulation of thousands of users concurrently accessing a web application [25]. Locust operates by launching multiple virtual user instances, or ”swarms”, to generate load on the target web application. These virtual users simulate real-world user behavior by sending Hypertext Transfer Protocol (HTTP) requests to the web server and measuring response times. With Locust, developers can define user scenarios, specifying actions such as HTTP requests, form submissions, or Application Programming Interface (API) calls, to accurately mimic user interactions [26].
In our study, the performance data obtained from the recruiting system will be integrated with the Locust test. By executing multiple test instances consecutively and at various load we aim to evaluate the performance and scalability of the recruiting system under different traffic conditions. This approach ensures the reliability of the test results by minimizing variability through consistent testing environments. Through this integration, we seek to

gain insights into how the recruiting system responds to varying loads, aiding in estimating CPU and RAM resources needed for different loads, which are key factors for choosing tiers and pricing structures on different CSPs.

3.4.2 Locust Metrics

The Locust Metrics section provides a detailed examination of the performance metrics obtained through load testing with the Locust tool. These metrics offer valuable insights into the behavior and responsiveness of the web application under varying levels of simulated user traffic. Key metrics captured during the load testing process include:

- **Response Time:** Response time measures the duration between sending a request to the web server and receiving a complete response. It indicates how quickly the application can handle user requests and is crucial for assessing overall user experience.
- **Requests Per Second** This metric indicates the number of requests that the server handles per second. It is a key performance indicator that reflects the application's throughput under load. Higher values suggest better performance, implying that the server can handle more interactions concurrently.
- **Error Rate:** Error rate quantifies the percentage of requests that result in errors or failures during the load test. It highlights potential issues such as server overload, resource constraints, or software bugs that may impact the application's reliability and availability.

3.4.3 CSP Monitoring Tools

To complement the data collected through Locust, built-in monitoring tools provided by each cloud service provider were employed. These tools are essential for gathering system-level performance metrics, offering a comprehensive view of the infrastructure's behavior during testing:

- **AWS CloudWatch**
- **Azure Monitor**
- **Google Monitoring**
- **Heroku Monitors**

Each tool was configured to record the following metrics in real-time:

- **CPU Utilization (%):** Indicates the percentage of allocated compute resources being used, reflecting the system's capacity to handle increased loads.
- **Memory Used (%):** Monitors the memory usage, critical for identifying potential bottlenecks in data processing and management.

3.4.4 Data Collection Process

To ensure a thorough evaluation of cloud provider performance under simulated load, the data collection process was structured into four key phases: setup and configuration, execution of load tests, data aggregation, and data analysis. Each phase was designed to capture and analyze both application-level and infrastructure-level metrics, providing a comprehensive overview of system behavior.

1. **Setup and Configuration:** Before initiating the load tests, each cloud provider's monitoring tool was configured to track the specified metrics. This setup ensured that once the tests began, data collection would be instantaneous and synchronized with the activities simulated by Locust.
2. **Execution of Load Tests:** As the Locust tasks were executed, both application-level and infrastructure-level metrics were collected simultaneously. The tasks generated traffic that mirrored real user behavior, triggering the cloud monitoring tools to capture data reflecting the system's response to the load.
3. **Data Aggregation:** Data from Locust and the cloud monitoring tools were aggregated to provide a unified view of each test scenario. This aggregation facilitated a comprehensive analysis, correlating application performance metrics with underlying system metrics.
4. **Data Analysis:** The collected data was then analyzed to draw insights into each cloud provider's performance, scalability, and reliability. This analysis helped identify the strengths and weaknesses of each cloud setup, guiding recommendations for optimization and resource allocation.

This methodical approach to data collection ensured that every aspect of the cloud service performance was monitored and recorded, providing a

robust dataset for subsequent analysis. The integration of Locust with native cloud monitoring tools offered a detailed perspective on how different cloud environments react under similar operational stresses.

3.4.5 Execution of Load Tests

During the execution phase of load testing, specified virtual machine configurations are systematically evaluated under incrementally increasing loads to assess their performance limits and operational robustness. Each of these tests is conducted in a controlled and isolated environment on our own servers. Key elements of this phase include:

- **Atlas Specifications:** 2 vCPUs and 2 GiB of memory
- **Virtual Machine Specifications:** start with 1 vCPU and 1 GiB of memory
- **Incremental Load Testing:** The load is gradually increased, starting from a low rate of RPS to higher levels until the system either fails or exhibits unacceptable response times.
- **Termination of Test Criteria:**
  - Avereage response > 1000 ms
  - CPU utilization > 80%
  - Memory Used > 10%
- **Post-Test Adjustments:**
  - Clear the database to ensure each test starts with the same conditions.
  - Terminate processes and monitor until metrics stabilize.

3.5 Planned Data Analysis

This section outlines the methodologies employed for analyzing the data collected in this study, which comparing pricing, performance and deployment complexities in different CSPs. To ensure a comprehensive evaluation, an array of analysis methods will be applied.

3.5.1 Statistical Data Analysis

We structured our data analysis approach to offer a comprehensive and comparative review of the collected performance and pricing data. In this process, Minitab [27] will be used for data input, graph plotting and statistical analysis. Using standard plotting techniques, separate plots for each metric on the diffrent cloud service providers will be created. This method facilitates a detailed examination of variation patterns across different metrics.
To improve the clarity of comparison between the CSPs, the mean for each metric will be calculated and used. These means were then presented in a table, and the differences were visualized as percentages in a bar graph. This visual representation simplified the comparison of individual metrics, providing a clear overview of the disparities between the platforms.

3.5.2 Qualitative Data Analysis

In order to get a comprehensive understanding of the detailed nuances involved and to conduct a thorough evaluation of the CSPs, qualitative analysis will be used based on the observations and insights mostly gained from practical work during the case study. It provides a straightforward overview of data patterns and explores deeper insights into real-world problems [28]. This will be used to present and analyse pricing data and deployment complexities to diffrent CSPs. The analysis also focuses on deployment efficiency and compatibility with existing systems. Findings reveal if the CSPs offer user-friendly interfaces and streamlined deployment processes. Additionally, checking if integrating new cloud services with existing infrastructure presenting challenges considering configuration to ensure compatibility with cloud service providers.

3.6 Evaluation framework

To accurately assess and compare the cost of various CSPs, our evaluation framework is designed around key metrics that reflect both the system's capacity and user interaction quality. The primary tools utilized for capturing these metrics are the CSPs' monitoring tools and the Locust framework. The metrics collected are described in Sections 3.4.3 and 3.4.2. By collecting these metrics we can determine the operational limits of Northabs application on different CSPs and to identify when an upgrade is necessary due to insufficient resources or poor user experience. This understanding will enable us to make

informed decisions regarding scaling up resources or switching between CSPs based on cost-effectiveness and performance efficiency.
Storage costs were not included in our comparative analysis. All cloud providers under consideration utilize Atlas MongoDB, which offers a consistent pricing model across different platforms. Since the storage requirements and costs are uniform irrespective of the chosen CSP, there is no competitive advantage or disadvantage to be gained from this metric.

Chapter 4

Implementation

This chapter provides a detailed exploration of the methodologies, tools, and materials employed throughout the evaluation process. In order to assess the performance of different CSPs, we employed different tools and analytical methods. Our methodology started with an extensive literature review focusing on cloud computing and its different providers. This foundational step allowed us to gain insights into the broader landscape of cloud technologies, understand key concepts, and familiarize ourselves with the diverse range of services offered by various CSPs. Following this, we started on the pivotal task of selecting which CSPs to evaluate further. This decision was steered by the specific requirements of the recruitment system, which served as our primary use case. By aligning the selection process with the needs and intricacies of our system, we aimed to ensure that our evaluation would be both relevant and actionable.
Once the CSPs were identified, our focus turned towards the collection of data and the analysis of pertinent factors such as cost efficiency and deployment complexities. Through a systematic approach, we aimed to gather comprehensive data sets that would facilitate a robust and insightful evaluation of each provider's performance within the context of our recruitment system.

4.1 Selection of Cloud Service Providers

The decision regarding which CSPs to further evaluate was based on the specific needs of the recruitment system, which acted as our primary use case. The general criteria for CSP selection revolved around offering businesses flexibility, scalability, and cost-effectiveness. Ultimately, our focus narrowed down to the major cloud service providers AWS and Azure for assessing IaaS,

and Heroku and GCP's App Engine for PaaS evaluation.
Acquiring comprehensive information on the diverse services offered by the CSPs mentioned involved several steps. Initially, we investigated the services available from virtual machines and their specification to their security features. Subsequently, we conducted a thorough analysis of the pricing structures associated with each service, including on-demand rates, reserved instances, and spot instance pricing models. Concurrently, we collected data on the deployment complexities inherent in each service, encompassing setup procedures, configuration requirements, application deployment processes, and ongoing service management tasks. Furthermore, the gathered data encompasses not only comparisons against the CSP alternatives but also against real-world cost data derived from the recruitment system, recorded in Minitab to facilitate a holistic overview of the offerings from different providers.

4.2 Implementation of Locust Tests

The load testing environment was implemented using Locust, an open-source load testing tool. Locust allows for the simulation of concurrent users making HTTP requests to test the scalability and performance of web applications and services hosted by various cloud providers. The implementation involved creating a set of sequential and random tasks mimicking real user interactions with the application's endpoints.

4.2.1 Script Overview

The Locust test script is structured as a series of tasks within a class inheriting from TaskSet, a built-in class from Locust that facilitates the modeling of complex user behaviors. In this context each task represents a specific user behavior, where a user behaviour and a task are interactions with the system ranging from registering a user to applying for a job. These tasks are aimed at evaluating the system's responsiveness and robustness under load, effectively simulating the variety of actions a real user might perform. The key tasks implemented are as follows:

- **Register Company:** This task involves making a POST request to register a company with a unique name and a standard description. Successful registrations append the company ID to a dataset for later use.

- **Create User:** Depending on available company IDs, this task creates a user associated with a randomly selected company, simulating the user registration process.
- **Login User:** Users attempt to log in using their credentials. Successful logins capture and store authentication tokens for subsequent requests requiring authentication.
- **Create Job Listing:** Authenticated users post job listings, requiring a valid token. This task tests the application's ability to handle authenticated write operations under load.
- **Get Company Details:** Fetches details for a randomly selected company, testing the read operations' responsiveness.
- **Apply for Job:** Simulates a user applying for a job, including uploading a resume. This complex task tests the system's ability to handle multipart/form-data requests and the robustness of job application features under stress.
- **Retrieve All Job Listings:** A simple GET request to retrieve all job listings, testing the system's capacity to handle large-scale read operations.

The script integrates with a custom DataHandler class to manage data persistence between tasks. This setup enables tasks to use data generated by other tasks, mimicking real-world user interactions where subsequent actions depend on the outcomes of previous ones.
Each task uses a context manager to handle the response. If the request is successful, the script logs positive feedback; otherwise, it logs a failure message. This method helps in identifying endpoints that underperform or lead to errors under stress conditions.

4.2.2 Code snippet

In this subsection, we delve into an example of the Locust test code used in our thesis to simulate user interaction with a job application system hosted on various cloud service providers. This specific code snippet illustrates how a simulated user applies for a job, an action that tests the application's ability to handle form submissions, file uploads, and database interactions under load.

Listing 4.1: Apply for job code in Locust

```
@task
def apply_for_job(self):
    job_listing_ids = self.jobs.load_data()
    user_emails = self.users.load_data()
    if not job_listing_ids or not user_emails:
        print("Jobs or users list is empty.")
        return

    job_listing_id = random.choice(job_listing_ids)
    user_email = random.choice(user_emails)
    first_name = "TestFirstName"
    last_name = "TestLastName"
    phone = "1234567890"

    with open('cv.txt', 'rb') as cv_file:
        files = {
            'cv': cv_file,
        }

        data = {
            "jobListingId": job_listing_id,
            "createdDate": "2024-04-16T08:00:00Z",
            "firstName": first_name,
            "lastName": last_name,
            "email": user_email,
            "phone": phone,
        }

        with self.client.post("/jobapplication/apply", data=data,
        files=files, catch_response=True) as response:
            if response.ok:
                print("Applied for job {job_listing_id}
                with user {user_email}")
            else:
                response.failure("Failed to apply for job
                {job_listing_id} with user {user_email}")
```

1. **Data Loading:** The task starts by retrieving lists of job listings and user

emails from the database, which are stored in local data handlers. This simulates the real-world scenario where a user selects from available job listings to apply.

2. **Random Selection:** It randomly selects one job listing and one user email to simulate the application process. This randomness is representative of various users interacting with the application in unpredictable ways.

3. **File Handling:** The task handles file operations where it opens a resume file **'(cv.txt)'** in binary read mode. This file represents the resume that the user would upload as part of the job application.

4. **Form Submission:** A POST request is made to the job application endpoint with the user's details and the resume file.

5. **Response Handling:** The script checks the response from the server. If the request is successful, it logs the successful application. If not, it logs a failure, which helps in identifying issues with the application process under load conditions.

Similar to the **apply_for_job()** task, other tasks within the Locust script follow a parallel structure, where:

- User behaviors are defined to interact with different endpoints, like registering companies, creating users, or logging in.
- Data is dynamically loaded and utilized within each task to ensure variability and realism in the simulation.
- Responses are handled to log successes and identify failures, crucial for stress testing the robustness of the application.

4.3 Implementation using IaaS

This section explores the deployment of an Express.js backend on two major cloud service providers: AWS and Microsoft Azure. Each deployment utilizes Ubuntu Server 24.04 LTS, tailored to the specific virtual machine configurations and cloud services provided by each platform. The ensuing subsections will detail the setup and configuration stages for each, emphasizing secure SSH access, network security settings, and the installation of necessary software components.

4.3.1 Instance Setup

Setting up a VM is the foundational step in deploying applications in the cloud. This involves selecting an appropriate VM image and size based on the application requirements and budget constraints. Both AWS and Azure offer similar VM configurations with 1 vCPU and 1 GiB of RAM.

4.3.1.1 AWS Instance Setup

On AWS, the deployment process began with the creation of an Amazon EC2 instance. Ubuntu Server 24.04 LTS was selected for its reliability and long-term support, which ensures regular security updates and system stability. The chosen instance type was t2.micro, recognized for its cost-effectiveness and ability to handle small to medium-sized applications with moderate user traffic. This instance type provides a balanced mix of CPU, memory, and network resources, making it an ideal choice for projects with limited computational demands but potential for growth.

4.3.1.2 Azure Instance Setup

For Azure, the setup involved provisioning a virtual machine using the same Ubuntu Server 24.04 LTS. The selected instance type was a standard B1s, which mirrors the t2.micro in terms of specifications with 1 vCPU and 1 GiB of RAM. This instance type is also tailored for cost-sensitive scenarios, offering a flexible CPU usage plan that allows the VM to accumulate CPU credits when idle. These credits can be utilized to handle spikes in demand without incurring additional costs. This feature makes the B1s particularly effective for projects where workload performance requirements vary, ensuring efficiency and cost control.

4.3.2 Configuring SSH Access

Secure Shell (SSH) access is critical for managing cloud servers. This protocol ensures secure file transfers and remote logins over insecure networks.

4.3.2.1 AWS SSH Configuration

AWS facilitates secure SSH access by allowing users to create and manage key pairs directly through its management console. This key pair is then associated with the EC2 instance at launch. The private key file, which is downloaded by the user, must be carefully secured and correctly referenced when initiating

SSH connections to ensure secure and encrypted communication with the instance.

4.3.2.2 Azure SSH Configuration

Similarly, Azure supports SSH key pair generation during the VM creation process or allows users to upload an existing public key. Once the VM is deployed, the public key is stored, and the corresponding private key is used for SSH sessions. Azure also emphasizes the importance of securing the private key and provides additional options for enhancing SSH security through its platform features like Azure Security Center.

4.3.3 Network Security Configuration

Configuring network security involves setting up rules that control inbound and outbound traffic to ensure that the VM is protected from unauthorized access.

4.3.3.1 AWS Network Security

AWS utilizes security groups to define firewall rules that govern the traffic to and from EC2 instances. For this project, rules were specifically crafted to allow only HTTP (port 80) and Hypertext Transfer Protocol Secure (HTTPS) (port 443) traffic from known and safe Internet Protocol (IP) addresses. This setup restricts server access to legitimate web traffic, thereby mitigating potential security threats.

4.3.3.2 Azure Network Security

Azure employs Network Security Groups (NSGs) to manage traffic to VMs. Like AWS, specific rules were implemented to permit only HTTP and HTTPS traffic. Additionally, Azure offers integration with Azure Monitor and Azure Security Center, which provide advanced monitoring and threat detection capabilities, enhancing the security posture of the VM.

4.3.4 Environment Setup

Once the virtual machines were securely accessible, the next step involved preparing the software environment for deploying the Express.js backend. This setup process was consistent across both AWS and Azure platforms, focusing on the installation of Node.js and Node Package Manager (NPM).

Node.js is crucial for server-side execution of the backend, while NPM, the Node Package Manager, handles the management of the application's dependencies. This preparation ensures that the backend can efficiently manage package installations and updates, which is vital for the ongoing maintenance and scalability of the application.

4.3.5 Deployment of the Application

The deployment process for the Express.js application followed a structured approach on both cloud platforms. Initially, the application's source code was securely transferred to the virtual machine using secure file transfer protocols. Following this, NPM was used to install all necessary dependencies as specified in the application's package.json file.
Environment variables were set to configure the application properly for the production environment. These included necessary configurations such as database connection strings, API keys, and other sensitive data required for the application to function correctly.

4.4 Implementation using PaaS

This section delves into the deployment of an Express.js backend on two leading PaaS providers: Heroku and GCP's App Engine. Each deployment is tailored to the unique capabilities and tools offered by the respective platforms. The subsequent subsections will detail the deployment processes for each, emphasizing the use of Heroku's integrated Git-based workflows and GCP's App Engine containerization approach

4.4.1 Deployment on Heroku

In the initial phase of deploying northabs application to Heroku, the deployment process was facilitated by the Heroku Command Line Interface (CLI), which seamlessly integrates with the Git version control system. This integration streamlined the deployment process, enabling seamless updates and effective scaling. Initially, the Heroku CLI was installed, followed by executing **"Heroku login"** to authenticate access to Heroku services through a web browser.
Once authenticated, a Heroku application was created within the project directory. This was accomplished by initializing a local Git repository if not already present using **"git init"**, and linking it to Heroku with the command

”heroku git:remote -a your-app-name”. This command sets up a new remote, known as 'heroku', which points to the Heroku application.
Before deployment, it was ensured that the **”package.json”** file was correctly set up to define the start script and dependencies. Additionally, environmental variables from the **”.env”** file were transferred to Heroku's configuration using Heroku's config vars. This was achieved by using **”heroku config:set KEY=VALUE”** for each variable, ensuring that the environment of the deployed application mirrors the local development environment.
The local changes were then staged using **”git add .”**, and committed with **”git commit -am ”msg”**. Deployment was initiated by pushing these changes to the Heroku remote with **”git push heroku master”**. Heroku's build system automatically detected the Express.js application, installed the required dependencies, and prepared the environment for launching the app.
To confirm the deployment, **”heroku open”** was executed, which launched a web browser to view the deployed application. For ongoing monitoring and troubleshooting, real-time logs were accessed via **heroku logs –tail**, allowing for immediate feedback and operational insights. Through these steps, the Express.js backend was effectively deployed to Heroku.

4.4.2 Deployment on GCP App Engine

For the deployment of the Express.js backend on Google Cloud Platform's App Engine, the approach required encapsulation of the application within a Docker container. This process began with the creation of a Dockerfile in the root of the project directory. The Dockerfile defined the environment on which the Express.js backend would run, specifying the base image, in this case, an official Node.js image, which provides a stable foundation for Node.js applications.
The Dockerfile also outlined the steps for copying the application source code into the container, installing dependencies via NPM install, and defining the command to start the application. Here is a brief rundown of the key Dockerfile directives used:

- **”FROM node:latest:”** This line sets up the Node.js environment.
- **”WORKDIR /usr/src/app:”** Specifies the working directory inside the container.
- **”COPY . .:”** Copies the local project files into the container.
- **”RUN npm install”:**Installs the project dependencies.

- **”RUN npm run build”:** Compile TypeScript to JavaScript
- **”EXPOSE 8080”:**Make port 8080 available to the world outside this container
- **”CMD [”node”, ”dist/app.js”]”:** Defines the command that runs the app.

Once the Dockerfile was prepared, the next step involved configuring the GCP App Engine environment through an app.yaml file, which specifies runtime configuration, including scaling and versioning policies. See listing 4.2 for the app.yaml file.
The deployment to GCP's App Engine was finalized by executing **”gcloud app deploy”** with Google Cloud Software Development Kit (SDK), which pushed the Docker container to the cloud, handled the build process on GCP's infrastructure, and deployed the containerized application to a dynamically managed environment. This command streamlined the deployment process, abstracting much of the complexity associated with managing individual server instances.

Listing 4.2: app.yaml file for GCP

```
runtime: custom
api_version: '1.0'
env: flexible
threadsafe: true

manual_scaling:
  instances: 1

resources:
  cpu: 1
  memory_gb: 1
  disk_size_gb: 10

liveness_check:
  initial_delay_sec: '300'
  check_interval_sec: '30'
  timeout_sec: '4'
  failure_threshold: 4
  success_threshold: 2
```

```
readiness_check:
  check_interval_sec: '5'
  timeout_sec: '4'
  failure_threshold: 2
  success_threshold: 2
  app_start_timeout_sec: '300'

service_account: name
```

Chapter 5

Results and Analysis

In this chapter, a comprehensive exploration of the study's outcomes, including an in-depth review of the different cloud service providers will be revealed. Delving into the objectives of the study and presenting a detailed account of the research findings. This entails a thorough analysis of documentations, price lists, collection of deployment data, and the results of the case study to evaluate the performance, availability, scalability, pricing, and deployment complexities of the diffrent CSPs. Additionally, we explore the parts related to data analysis and presenting the methods we used to examine the collected data from the reviews and deployment tests. Details about studying the data through statistical analyses, focusing on relevant parameters and metrics . We also explore descriptive data analysis to illustrate the insights we gained from our thorough investigation related to deployment complexities.

5.1 Pricing Structures and Models

In this section, we thoroughly explore the pricing structures of various CSPs, crucial for organizations aiming to optimize costs while using cloud services effectively. Pricing structures and models vary between the selected cloud service providers. The price lists of each CSP will be presented, detailing the diverse pricing tiers, models, and options available.

5.1.1 AWS

AWS offers diverse pricing models. Users can choose from on-demand pricing, spot instances, dedicated hosts, reserved instances, and savings

plans. Each model has its pros and cons, ranging from flexibility and cost-effectiveness to long-term commitment requirements and potential complexities in management.

We have chosen to proceed with on-Demand Pricing model for Amazon EC2 because it is an IaaS that offers flexibility and ease without long-term commitments, making it ideal for variable workloads and rapid scaling.The On-Demand Pricing model for Amazon EC2 instances offers flexibility and cost-efficiency by allowing users to pay for compute capacity on an hourly or second basis, with no long-term commitments. This eliminates the need for upfront hardware planning, purchasing, and maintenance costs, transforming large fixed expenses into smaller variable costs. Pricing for On-Demand Instances is calculated per instance-hour consumed, from the instance launch until termination or stopping. Partial instance-hours are billed per second for Linux, Windows, and Windows with SQL Instances, and as a full hour for other instance types.

The pricing includes costs for running both private and public Amazon Machine Images (AMIs) on various OSes, including Windows Server and Linux distributions. Additionally, there are separate pricing options available for instances running Microsoft SQL Server, SUSE Linux Enterprise Server, and Red Hat Enterprise Linux (RHEL).

Instance name ▲	On-Demand hourly rate ▽	vCPU ▽	Memory ▽	Storage ▽	Network performance ▽
t2.nano	$0.0058	1	0.5 GiB	EBS Only	Low
t2.micro	$0.0116	1	1 GiB	EBS Only	Low to Moderate
t2.small	$0.023	1	2 GiB	EBS Only	Low to Moderate
t2.medium	$0.0464	2	4 GiB	EBS Only	Low to Moderate
t2.large	$0.0928	2	8 GiB	EBS Only	Low to Moderate
t2.xlarge	$0.1856	4	16 GiB	EBS Only	Moderate
t2.2xlarge	$0.3712	8	32 GiB	EBS Only	Moderate

Figure 5.1: Range of available EC2 instances

Users can choose from a range of EC2 instance types as shown in Figure 5.1 with varying numbers of virtual CPUs (vCPUs), and custom vCPU counts can be specified at instance launch. The pricing model remains consistent regardless of the vCPU count.

5.1.2 Azure

Azure's pricing structure is influenced by factors like service type, required capacity, location, and management level. It offers a free tier, granting complimentary access to select services for the initial 12 months and permanent free usage for specific services. The primary pricing model is "pay as you go," charging based on actual usage. Azure also provides discounts for reserved instances (commitment of 1 or 3 years) and spot instances (virtual machines from spare capacity). This is part of an extensive Financial Operations (FinOps) guide series covering topics such as Azure Free Tier, pricing models, and savings options like Azure Hybrid Benefit and Dev/Test Pricing.
Azure's VM offerings encompass a diverse array of series, each designed to meet specific workload demands, as shown in Figure 5.2. The B-series, featuring a unique CPU credit model, enables burstable performance, ideal for fluctuating workloads.

Size	vCPU	Memory: GiB	Temp storage (SSD) GiB	Base CPU Performance of VM (%)	Initial Credits	Credits banked/hour	Max Banked Credits	Max data disks	Max uncached disk throughput: IOPS/MBps
Standard_B1ls2	1	0.5	4	5	30	3	72	2	160/10
Standard_B1s	1	1	4	10	30	6	144	2	320/10
Standard_B1ms	1	2	4	20	30	12	288	2	640/10
Standard_B2s	2	4	8	20	60	24	576	4	1280/15
Standard_B2ms	2	8	16	30	60	36	864	4	1920/22.5
Standard_B4ms	4	16	32	22.5	120	54	1296	8	2880/35
Standard_B8ms	8	32	64	17	240	81	1994	16	4320/50
Standard_B12ms	12	48	96	17	360	121	2908	16	4320/50
Standard_B16ms	16	64	128	17	480	162	3888	32	4320/50
Standard_B20ms	20	80	160	17	600	202	4867	32	4320/50

Figure 5.2: Range of available Azure VM models

We chose to proceed with the B-series because it is an IaaS with a unique CPU credit model, which allows us to leverage burstable performance capabilities while optimizing costs [29]. This flexibility is crucial for our workload, which experiences varying levels of demand throughout the day. With the B-series, we can efficiently manage our resources, ensuring that we have

the computing power we need when it's required, without overspending on constant high-performance instances. Additionally, the B-series enables us to scale seamlessly as our workload evolves, providing the agility necessary to adapt to changing business needs. Overall, the B-series VMs offer the ideal balance of performance, cost-effectiveness, and scalability, making them the perfect choice for our cloud infrastructure.

5.1.3 Heroku

Heroku's pricing models are flexible with a variety of pricing tiers, making it possible to scale up or down as needed. This flexibility is particularly beneficial for startups and small businesses, as it enables them to start small and gradually increase resources as their applications grow [30].
Heroku operates within lightweight, self-contained Linux containers known as "dynos." These dynos come in various types, each optimized to deliver optimal performance for different types of applications. One notable aspect of Heroku's pricing is its pay-as-you-go model. This means users only pay for the resources they consume, whether it's dynos for running applications, databases for storing data, or add-ons for extending functionality. This model provides users with fine-grained control over their spending and eliminates the need for long-term commitments or upfront payments.
Heroku offers a variety of dyno types to support apps of all sizes, from small-scale projects to high-traffic production services. Memory, CPU share, and other differentiating characteristics for each Common Runtime dyno type are listed in the Figure 5.3 below:

Plan	Memory (RAM)	CPU Share	Compute	Sleeps
Eco	512 MB	1x	1x-4x	✓
Basic	512 MB	1x	1x-4x	
Standard-1X	512 MB	1x	1x-4x	
Standard-2X	1 GB	2x	2x-8x	
Private/Shield-S	1 GB	100%	12x	
Performance/Private/Shield-M	2.5 GB	100%	12x	
Performance/Private/Shield-L	14 GB	100%	50x	
Performance/Private/Shield-L-RAM	30 GB	100%	24x	
Performance/Private/Shield-XL	62 GB	100%	50x	
Performance/Private/Shield-2XL	126 GB	100%	100x	

Figure 5.3: Range of available Heroku's dyno types

5.1.4 Google App Engine

Google Cloud Platform's pricing structure is designed to offer flexibility and cost-efficiency across a variety of service models, accommodating diverse computing needs and budget constraints. Like its competitors, GCP provides a free tier that includes limited access to many common services, allowing users to try before they buy. The primary pricing strategy is "pay as you go," which bills users based on their actual consumption of resources.

The pricing model for vCPU and memory in GCP is straightforward, with a per-core-hour and per-GB-hour rate respectively. This approach provides transparent and predictable pricing, allowing users to easily estimate their costs based on their resource utilization. For example, vCPUs are charged at $0.0579 per core hour and memory at $0.0078 per GB hour. See Figure 5.4.

GCP also offers different instance classes with fixed costs per hour, which can suit various application needs, from lightweight web servers to heavy data-processing applications. These instance classes range from the smaller B1 class at $0.05 per hour to the more robust B8 class at $0.4 per hour. Each class is tailored to match different performance and workload requirements, providing businesses with the flexibility to choose the most cost-effective configuration for their specific tasks. See Figure 5.5.

Instance class	Cost per hour per instance
B1	$0.05
B2	$0.1
B4	$0.2
B4_1G	$0.3
B8	$0.4
F1	$0.05
F2	$0.1
F4	$0.2
F4_1G	$0.3

Figure 5.4: Range of available App Engine standard environment pricing

Resource	Unit	Unit cost
vCPU	per core hour	$0.0579
Memory	per GB hour	$0.0078
Persistent disk	Priced as Compute Engine persistent disk, which is called `Storage PD Capacity` on your bill.	
Outgoing network traffic	Priced as Compute Engine internet egress	
Incoming network traffic	Gigabytes	Free

Figure 5.5: App Engine flexible environment pricing

5.2 Storage pricing

Based on the storage requirements of Northab's recruiting system and convinience, we have proceeded with storing the database in MongoDB's platform Atlas. MongoDB Atlas is a fully-managed cloud database service provided by MongoDB, offering developers and organizations a scalable, reliable, and secure platform for deploying MongoDB databases in the cloud. With features such as managed infrastructure, seamless scalability, high availability, global distribution, robust security, and integrated tools, MongoDB Atlas simplifies database management and enables the development of modern, cloud-native applications with confidence [31]. This process makes the cost of database storage constant for all the tests conducted on different cloud service providers. However, additional costs for storage on the cloud provider's platform are redundant.

- Storage within MongoDB: When you store data in MongoDB, you're utilizing storage within the database itself. MongoDB allocates storage space within its database system to store your data. The cost associated with this storage is typically included in the pricing structure of MongoDB itself or any managed MongoDB service you're using.

- Storage on the Cloud Provider: When you deploy MongoDB on a cloud platform and using VMs or managed database services provided by the cloud provider, these services require storage for various purposes such as VM disks, database backups, and snapshots. The cost associated with this storage is separate from the storage within MongoDB itself and is charged by the cloud provider.

5.3 Performance Analysis

This section outlines the performance metrics for various CSPs under identical testing conditions as detailed in Section 3.4.5. It includes metrics like CPU utilization, memory usage, and response times, providing insights into each CSP's efficiency and responsiveness. The results are derived using the Locust tool and CSP-specific monitoring tools, as described in Sections 3.4.2 and 3.4.3.

5.3.1 CSPs Instances

The specification for each CSP is the following:

CSP	Name	vCPU	GiB
AWS	t2.micro	1	1
Azure	Standard B1s	1	1
Azure	Standard B2s	2	4
GCP	Flexible Environment	1	1
Heroku	Standard 2X	Shared	1

Table 5.1: CSP Configurations

5.3.1.1 AWS t2.micro (IaaS)

RPS	CPU Utilization (%)	Memory Used (%)	Average Response Time (ms)
0	1	25	0
20	10	37	16.43
40	15.5	30.5	19.98
60	20.1	30.5	21.32
80	24.3	35.5	25.71
100	31.7	35.6	35.74
120	37.4	37.5	48.88

Table 5.2: Performance Metrics of Azure B1s at Various Load Levels

The AWS t2.micro instances demonstrate a steady increase in CPU utilization and memory use with RPS, as shown in Table 5.2. The response time remains below 50 milliseconds even at the highest load, indicating good performance under the specified conditions.

5.3.1.2 Azure B1s and B2s (IaaS)

RPS	CPU Utilization (%)	Memory Used (%)	Average Response Time (ms)
0	1%	70.7%	0
20	7.7%	81.35%	26.98
40	11.4%	87.99%	33.3
60	17.1%	92.97%	32.56

Table 5.3: Performance Metrics of Azure B1s at Various Load Levels

RPS	CPU Utilization (%)	Memory Used (%)	Average Response Time (ms)
80	10.3%	22.50%	35.65
100	12.9%	22.50%	43.25
120	15%	22.50%	50.11

Table 5.4: Performance Metrics of Azure B2s at Various Load Levels

Azure B1s instances start with a high baseline memory utilization of 70.7% and peak at 92.97% under load (Table 5.3). This high utilization required a manual upgrade to prevent crashes. However, in a production environment, it is typical to configure the Azure auto-scaler to automatically adjust resources when high memory usage is detected.
When upgrading to Azure's B2s the instance demonstrated more stable memory utilization around 22.50% across different load levels (Table 5.4).

5.3.1.3 GCP Flexible Environment (PaaS)

RPS	CPU Utilization (%)	Memory Used (%)	Average Response Time (ms)
0	5%	44.9%	0
20	20.5%	50.78%	101.33
40	34%	51.76%	112.59
60	52%	53.69%	128.65
80	64%	55.18%	140.9
100	75%	55.3%	163.1
120	80%	55.76%	199.35

Table 5.5: Performance Metrics of GCP Flexible Environment at Various Load Levels

GCP Flexible instances exhibit a sharp increase in CPU utilization and memory usage as load increases (Table 5.5). Notably, the response time escalates significantly beyond 80 RPS, pointing to possible performance limits under high load scenarios.

5.3.1.4 Heroku Standard 2X (PaaS)

RPS	Dyno Load Avg	Memory Used (%)	Average Response Time (ms)
0	0	5%	0
20	0.17	10.74%	144.21
40	0.27	11.52%	166.17
60	0.22	11.72%	209.01
80	0.27	11.9%	219.76
100	0.25	13.18%	249.72
120	0.25	13.8%	301.12

Table 5.6: Performance Metrics of Heroku Standard 2X at Various Load Levels

Figure 5.6: Herokus RPS and response time (ms) under stress

Heroku Standard 2X instances show relatively low memory utilization but exhibit significant performance instability under high loads, as depicted in Figure 5.6. The graphs illustrate pronounced dips in the RPS over brief periods. Each downturn in request throughput lasted approximately one minute, with these fluctuations cumulatively accounting for around four minutes of suboptimal performance within a ten-minute span.
This pattern of behavior highlights Heroku's difficulties in handling increased loads, leading to extended periods of heightened response times during observed spikes, which could impact user experience and service reliability.

5.3.2 Performance Comparison

This subsection compare the data presented in Section 5.3.1 to provide a analysis of the CSPs. Using the metrics detailed in Tables 5.2 through 5.6, it examines how each CSP performs relative to the others under similar operational conditions. The comparison highlights differences in CPU utilization, memory usage, and response times, offering insights into the efficiency and scalability of each provider.

5.3.2.1 RPS Range

CSP	Name	vCPU	GiB	RPS Range
AWS	t2.micro	1	1	0-120
Azure	Standard B1s	1	1	0-60
Azure	Standard B2s	2	4	60-120
GCP	Flexible Environment	1	1	0-120
Heroku	Standard 2X	Shared	1	0-120*

Table 5.7: RPS Capacity of CSPs

* Heroku could handle 120 RPS but had performance issues, as detailed in Section 5.3.1.4.

Table 5.7 indicates the RPS range of each CSPs. It is worth noting that Azure B2s and AWS t2.micro are likely capable of handling more than 120 RPS. This is evidenced by the performance metrics in Sections 5.3.1.1 and 5.3.1.2, where CPU utilization and memory usage for t2.micro were only 37.4% and 37.5%, respectively. For the B2s, these figures were even lower, at 17.1% for CPU utilization and 22.50% for memory usage.

5.3.2.2 CPU Utilization (%)

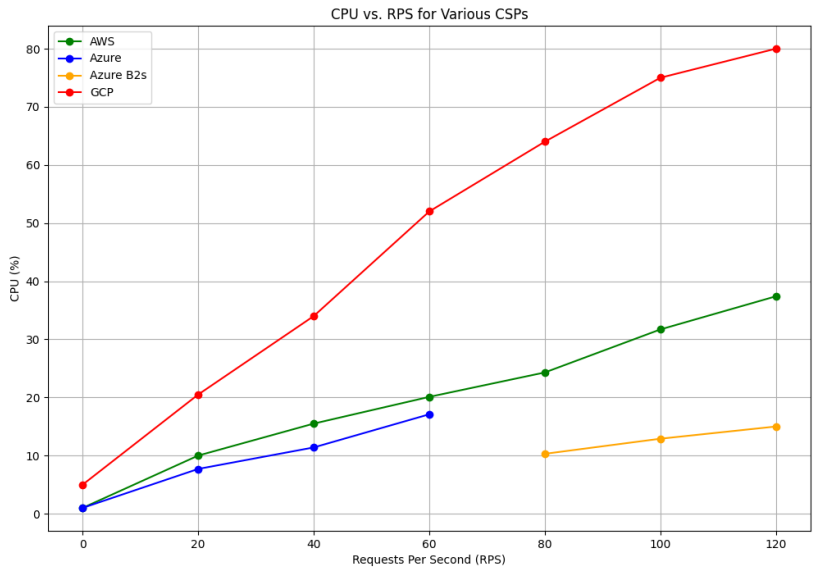

Figure 5.7: CPU Utilization (%) vs. RPS for CSP

The graph in Figure 5.7 demonstrates a nearly linear relationship between Requests Per Second and CPU utilization for each CSP analyzed. AWS and Azure display comparable levels of CPU utilization, with the notable distinction that Azure B2s instances employ 2 vCPUs compared to the typical single vCPU configuration. In contrast, GCP's App Engine exhibits the most pronounced increase in CPU utilization relative to RPS, indicating a steeper computational demand as request load escalates. It is important to note that Heroku is not included in Figure 5.7 because it does not report CPU utilization. Instead, Heroku provides metrics on average dyno load, which cannot be directly equated to CPU utilization.

5.3.2.3 Memory Used (%)

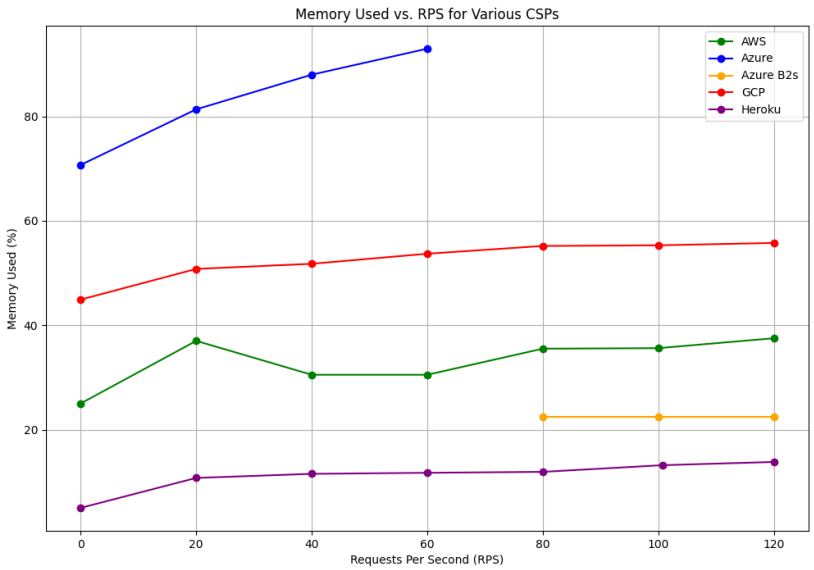

Figure 5.8: Memory Used (%) vs. RPS for CSP

The graph in Figure 5.8 reveals significant trends in memory utilization among various CSPs. Notably, the Azure VM, initially configured with 1 GiB of memory, starts at approximately 70% memory utilization. As demand increases, memory usage climbs toward 90%, indicating a need for an upgrade. After upgrading to 4 GiB of memory, the utilization stabilizes at just over 20%, showing a dramatic improvement in memory management. In contrast, the other CSPs begin at much lower memory usage percentages, suggesting that Azure's system inherently requires more memory compared to its competitors. With the exception of Azure, each CSP appears capable of handling around 100 RPSs without a significant increase in memory usage.

5.3.2.4 Average Response Time (ms)

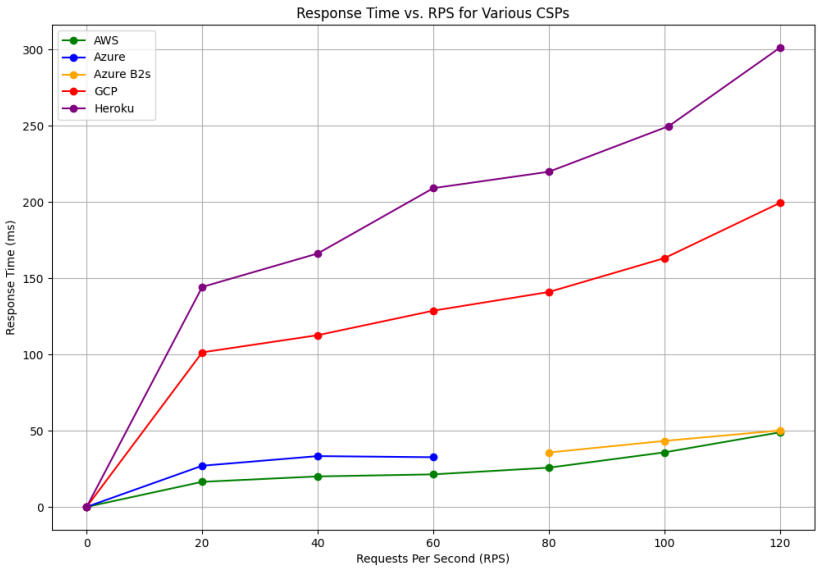

Figure 5.9: Response Time (ms) vs. RPS for CSP

In the graph presented in Figure 5.9, it is observed that AWS and Azure exhibit similar response times, which correlate closely with similar RPS levels. This similarity in performance can be attributed to the geographical location of their servers, both situated in Stockholm, Sweden. In contrast, GCP and Heroku do not offer the option to select this specific availability zone. For GCP, the nearest server location available is in Belgium. Heroku offers just two options limited to either the United States or Europe; for our purposes, we selected Europe. This variance in server location can significantly influence the response times observed for GCP's App Engine and Heroku compared to those for AWS and Azure.

5.4 Cost Analysis

The Table 5.8 provides a detailed comparison of the pricing models for AWS, Azure, Heroku and GCP. It focuses on three primary pricing strategies: Pay-as-you-go, Reserved Instances, and Subscription. Each model has been analyzed in terms of instance specifications, cost per hour, and yearly costs.

Pricing models ↓	Specifications	**AWS** (IaaS)	**Azure** (IaaS)	
Pay-as-you-go	Instance	t2.micro (1vCPU, 1GiB)	B1s (1vCPU, 1GiB)	B2s (2vCPU, 4GiB)
	Cost per hour	$0.0108 /hour	$0.010 /hour	$0.0416 /hour
	Yearly cost	**$94.608**	**$91.104**	**$364.416**
Reserved instance	Cost (1 year)	**$61.32 /year**	**$53.04 /year**	**$213.00 /year**
	Cost (3 years)	**$43.80 /year**	**$34.32 /year**	**$137.04 /year**
Subscription	Period	-	-	-
	Cost	-	-	-

Pricing models ↓	Specifications	**Heroku** (PaaS)		**GCP** (PaaS)
Pay-as-you-go	Instance	Standard-1X (1CPU, 512MB)	Standard-2X (2CPU, 1GB)	Flexible Instance (1vCPU, 1 GiB)
	Cost per hour	$0.035 /hour	$0.06 /hour	$0.063 /hour
	Yearly cost	**$306.6**	**$525.6**	**$551.88**
Reserved instance	Cost (1 year)	-	-	-
	Cost (3 years)	-	-	-
Subscription	Period	1 month	1 month	1 month
	Cost	**$25**	**$50**	**$45**

Table 5.8: CSP's Pricing Models: AWS, Azure, Heroku and GCP.

5.4.1 Pay-as-you-go

In the Pay-as-you-go model, AWS offers the t2.micro instance, which includes 1 vCPU and 1 GiB of RAM, at a rate of $0.0108 per hour. This equates to an annual cost of $94.608. Azure, on the other hand, the B1s instance with 1 vCPU and 1 GiB of RAM at $0.010 per hour, and the B2s instance with 2 vCPUs and 4 GiB of RAM at $0.0416 per hour. The yearly costs for these Azure instances are $91.104 and $364.416, respectively. This shows that while the B1s instance is slightly cheaper than AWS's t2.micro on an hourly basis, the B2s instance offers a higher capacity at a significantly higher cost.
In Heroku, the Standard-1X instance is priced at $0.035 per hour, leading to an annual cost of $306.6. The more powerful Standard-2X instance is priced at

$0.06 per hour, resulting in an annual cost of $525.6. GCP offers the Flexible Instance with 1 vCPU and 1.095 GiB of RAM at a rate of $0.063 per hour. This equates to an annual cost of $551.88. While GCP's Flexible Instance has a slightly higher cost per hour and yearly cost compared to Heroku's Standard-2X, it offers marginally higher memory capacity.

5.4.2 Reserved Instance

The Reserved Instance model offers substantial cost savings for long-term commitments, percentages are represented in Table 5.9. AWS's one-year reserved instance costs $61.32 annually, while a three-year reserved instance drops to $43.80 per year. Azure offers its B1s instance at $53.04 per year for a one-year commitment and $34.32 per year for a three-year commitment. For the higher-capacity B2s instance, the costs are $213.00 per year for a one-year reservation and $137.04 per year for a three-year reservation. This demonstrates that Azure provides more cost-effective solutions for longer commitments, particularly with the B1s instance, which is cheaper than AWS for both one-year and three-year reservations. Heroku and GCP do not offer distinct pricing for reserved instances. The focus for these providers is primarily on the Pay-as-you-go and Subscription models.

Name	Period	Yearly($)		Savings(%)	Upfront($)
		Pay-as-you-go	Reserved		
t2.micro	1 year	94.608	61.320	39.66%	61.320
t2.micro	3 years	94.608	43.800	56.90%	131.400
Standard B1s	1 year	91.104	53.040	39.59%	53.040
Standard B1s	3 years	91.104	34.320	60.82%	102.960
Standard B2s	1 year	364.416	213.000	42.10%	213.000
Standard B2s	3 years	364.416	137.040	62.76%	411.120

Table 5.9: Percentage of Yearly Cost Savings for Reserved Instances Compared to Pay-as-you-go with Total Upfront Cost

5.4.3 Subscription Model

Both Heroku and GCP provide a subscription-based pricing model. Heroku offers a one-month subscription for the Standard-1X instance at $25 per month and the Standard-2X instance at $50 per month. GCP's Flexible Instance is available at a monthly subscription cost of $45. This makes

GCP's subscription option moderately priced between Heroku's Standard-1X and Standard-2X offerings, potentially providing a balanced option for users seeking a subscription model. Both AWS and Azure do not offer a distinct subscription-based pricing model for these specific instances.

5.4.4 Comparison

The graph illustrated in Figure 5.10 present the yearly cost for each of the CSPs instances. The RPS range is based on our performance results from Section 5.3.1. Notably, the results unveiled some differences between IaaS and PaaS providers.

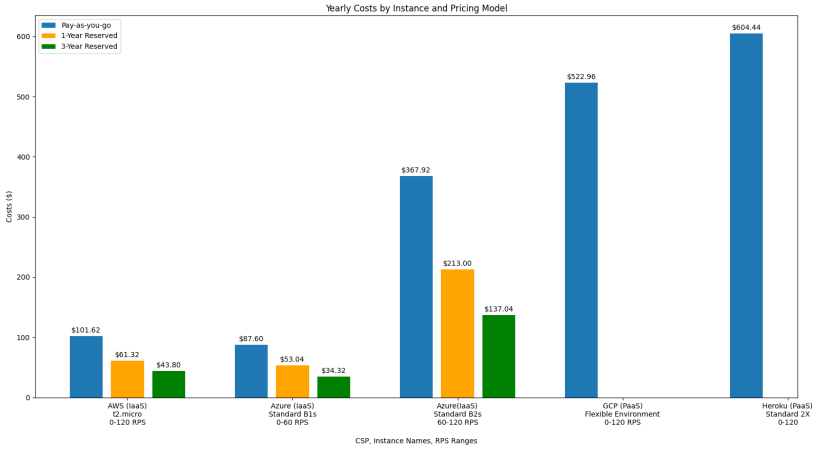

Figure 5.10: Yearly Costs by Instance and Pricing Model

In the IaaS domain, AWS emerged as a frontrunner, demonstrating remarkable cost efficiency as the recruiting system scaled. With AWS, no bottlenecks were encountered, and the cost remained relatively stable, regardless of the application size. On the other hand, Azure, a fellow IaaS provider, showcased a nuanced cost landscape. While initial costs aligned with expectations, a tipping point was discerned, necessitating an upgrade to higher-tier VM instances as demand surged. This nuanced scalability impacted cost efficiency, particularly for larger applications.
In PaaS, both Heroku and GCP offered compelling alternatives, despite distinctive nuances. Tests in both PaaS CSPs presented remarkable higher cost estimations than the IaaS. Heroku's platform, while facilitating rapid development and abstracting infrastructure complexities, presented a more

straightforward cost structure. However, as the application scaled, incremental costs accrued, underscoring the necessity of observant monitoring to avoid cost overruns. GCP, on the other hand, leveraged its flexible instance offerings to accommodate varying workloads seamlessly. The platform's cost efficiency persisted across different application sizes, epitomizing the allure of PaaS for scalable deployments.

5.5 Deployment Complexities

Deploying the application to different CSPs, involves tackling with the distinct challenges and advantages of each platform. Table 5.10 provides a comprehensive evaluation of deployment complexities across four major CSPs: AWS, Azure, Heroku, and GCP.

Table 5.10: Evaluation of Deployment Complexities Across CSPs

Metric	**AWS** (IaaS)	**Azure** (IaaS)	**Heroku** (PaaS)	**GCP** (PaaS)
Documentation	Offers a big variety of services with detailed resources.	Has simple and structured documentation for easy navigation.	Clear guidance for deploying, managing, and scaling applications.	Provides detailed and in-depth guidance for deploying.
Deployment Time	Complex due to many options and services and the learning curve that comes along.	Similar to AWS, flexible but can be time consuming due to many options and management.	Simplified deployment due to minimal management of the infrastructure.	Efficient deployment facilitated by managed infrastructure, minimal setup required.

Continued on next page

Table 5.10 – *Continued from previous page*

Metric	AWS	Azure	Heroku	GCP
Authorization	User-defined roles with unique permission controls for security.	A single source for permission management and authorization, access to each account has to be configured.	Requires 2FA and uses the Heroku CLI for secure interactions with the platform.	Requires 2FA and uses Google Cloud SDK for managing deployments and permissions.
Serverless API	EC2 requires management and provisioning of the environment [32].	Requires managing and maintaining the VM, including tasks such as provisioning, patching, scaling, and monitoring [33].	Serverless, abstracts away much of the infrastructure management [12].	Serverless, scales resources based on traffic, optimizing performance and cost efficiency [12].
Networking	Users can build private networks using AWS' Virtual Private Cloud (VPC) [34].	Relies on a virtual Private Network (VPN) [35].	Lacks complex networking infrastructure, yet offers communication via HTTP/HTTPS and Private Spaces [36] .	Simple built-in networking capabilities.

Continued on next page

Table 5.10 – *Continued from previous page*

Metric	AWS	Azure	Heroku	GCP
Logging and monitoring	Logs model metrics via CloudWatch, requires configuration.	Uses Azure Monitor and Application Insights for logging and performance monitoring.	Captures all logs with integrated logging facilities; manageable via Heroku CLI or third-party add-ons.	Utilizes Google Cloud Operations for comprehen-sive monitoring, logging, and diagnostics.
Scalability and Performance	Offers flexibility with a wide range of scalable resources and automated tools. Provides robust performance stability across its broad global infrastrucutre.	Excels in hybrid environments and integrates seamlessly with Microsoft technologies. Also offers tools for both horizontal and vertical scaling.	Provides user-friendly and quick scaling with a small set of specifications. But faces performance and cost challenges at larger scales.	Robust infrastructure supporting both vertical and horizontal scaling; includes auto-scaling and advanced load balancing.
Cost and Pricing Complexity	EC2 charges by the hour using a pay-as-you-go model and reserved instance discounts.	VM is charged by the minute and offers both pay-as-you-go and savings plans.	Charges based on dynos, not infrastructure usage. Dynos are containers running specified commands. Subscribtion available.	Follows a serverless billing model, charging based on resource usage rather than infrastructure provision.

In order to gain a comprehensive understanding of the detailed nuances involved and to conduct a thorough evaluation of the CSPs, we have undertaken a process of rating each metric across every CSPs based on our firsthand experiences with deployment, management, and monitoring. The results are presented in Table 5.11. Our approach involved actively engaging with the platforms, deploying applications, managing their operations, and closely monitoring performance metrics. These ratings are based on our direct observations and insights gained from practical usage during the case study.

Table 5.11: Rating of Deployment Complexities Across CSPs

Metric	**AWS** (IaaS)	**Azure** (IaaS)	**Heroku** (IaaS)	**GCP** (IaaS)
Documentation	7	8	9	7
Deployment Time	6	6	9	7
Authorization	5	6	8	8
Serverless API	-	-	10	10
Networking	7	7	7	7
Logging and monitoring	6	8	9	9
Scalability and Performance	9	7	6	7
Cost and Pricing Complexity	7	7	9	7
Score	**47**	**49**	**67**	**62**

5.6 Evaluation of CSPs

An analysis of different CSPs revealed unique features, especially in terms of pricing and deployment complexities. A detailed comparison revealed considerable price variations, emphasizing the importance of careful assessment to ensure cost-effectiveness. Evaluating pricing structures enables informed decisions aligning with specific business needs. Furthermore, the tests conducted on various application scales demonstrated how costs could fluctuate. Moreover, delving into deployment procedures highlighted differences in setup difficulty, resource distribution, and integration methods. These differences underscore the challenges organizations might encounter when configuring and rolling out cloud infrastructure across multiple providers.

5.7 Assessing reliability and validity of the data collected

5.7.1 Validity

The study's validity is supported by the data sources and the metrics chosen to evaluate the different CSPs. key metrics such as response time, resource utilization, cost and scalability significantly impact cloud performance and optimizing any cloud-based infrastructure effectively [37]. This data collected by Locust [23] represents valid and components that are comparable using diffrent CSPs.
The pricing data supports the validity of the study since its collected from the official websites of each CSP. The time span of the study also supports its validity since the tests are done in a reasonable time frame that align with the same or similar conditions for doing the tests, minimizing condition changes resulting by updated features and new technology.

5.7.2 Reliability

To ensure reliability, we adopted an approach that consisted of simulating user behavior on web applications to test how the system behaves under various levels of loads at consistent intervals. We utilized identical testing devices and protocols for all CSPs to standardize the process, minimizing the influence of random fluctuations and measurement discrepancies. This standardized approach reinforces the credibility and dependability of our findings. However, it's important to acknowledge the study's limitations, including constraints related to range, data availability, and methodology. These limitations should be considered when interpreting the results and drawing conclusions.
The study will only include a selection of few services offered by the selected CSPs, meaning that many services available may not be tested or compared. As a result, the study may not fully account for the diversity of cloud service offerings available in the market. The data utilized in this thesis is current as of May 2024. All analysis and conclusions drawn are based on data available up to this date. Furthermore, it's important to acknowledge that cloud services' performance can be affected by a multitude of factors beyond our control, such as network conditions, load on testing devices or CSP-specific updates. Measures are taken to minimize such effects by implementing strategies such as incorporating margins for observed results and theoretical specifications.

Nonetheless, complete elimination of these sources of variability is not possible. Additionally, although common metrics are used in this study, they might not give us sufficient information. Although we trust that our findings are consistent based on our chosen methods and metrics, further research using various methods would further confirm and enhance these results.

Chapter 6

Conclusions and Future work

Cloud service providers offer a variety of services with diverse pricing structures to meet evolving client needs. Each cloud service model provides different levels of control and management, presenting a challenge for companies to select the most suitable option.

This study addresses the challenge of selecting the optimal cloud service for a digital solution, proposing a decision-making approach based on test analysis of cloud price collection from four major providers. These collections are characterized by CPU, memory, and loads, which are key factors influencing cloud prices. Load testing these parameters based on size allows for the determination of average costs across all providers.

The results provide estimated pricing perceptions across cloud computing services and offer valuable insights for decision makers. The proposed approach integrates cost considerations and deployment complexities into the decision-making process, highlighting the importance of these parameters as a primary factor in cloud service selection.

Among the significant findings is that the grouping results align with the pricing policies of leading CSPs. Additionally, cost analysis findings are used to calculate the overall computing expenses of deploying a case study application across different cloud services, facilitating cost-based decision-making.

However, this study has limitations. It only considers compute costs for CPU, RAM, and load, excluding additional costs such as operation and security. Furthermore, the comparison of deployment models is based on provisioned capacity, overlooking other resource utilizations. The choice of the case study may be considered simplistic, as resource requirements scale linearly with workload demand. Additionally, the use of random requests to test the dataset

limits the study's accuracy, suggesting the potential for future improvements with more precise data traces.

6.1 Overview of findings

6.1.1 Pricing

Pricing models vary across different CSPs, but the most common models include pay-as-you-go, subscription, and reserved instance. These models specify how users are billed for utilizing cloud resources, shaping cost optimization strategies. AWS utilizes a flexible pay-as-you-go pricing model across a broad range of services, including compute power, storage solutions, and managed databases. Users pay only for the resources they consume, without upfront costs or long-term commitments. AWS also offers Reserved Instances, providing good discounts for users reserving cloud capacity for one or three years, which is suitable for businesses with predictable workloads and have a will to commit. Azure follows a similar pay-as-you-go approach, offering discounts through Reserved Instances and Hybrid Benefit options, particularly advantageous for stable, long-term cloud computing needs and Windows-centric environments, respectively. Heroku's pricing is based on dyno usage, add-ons, and data services, offering a transparent, scalable and on-demand model for diverse application needs. Heroku also offers subscription-based pricing, which charges a fixed monthly fee rather than billing by the hour. This model provides predictable costs but might not align with the variable usage patterns of some applications. Finally, Google Cloud Platform introduces similarly uses the pay-as-you-go and subscribtion-based models.

6.1.2 Deployment Complexities

Deploying an application, whether on IaaS platforms like AWS and Azure or PaaS providers such as Heroku and GCP's App Engine, entails a multitude of complexities. From selecting the right virtual machine configuration to configuring SSH access and network security, each step demands meticulous attention to detail. Moreover, transitioning between IaaS and PaaS environments introduces additional challenges, such as containerization for GCP's App Engine deployment. Throughout the process, maintaining security, ensuring scalability, and optimizing performance are paramount. Deploying on IaaS platforms like AWS or Azure has shown to offers a wider control over infrastructure components but demands

expertise and skills in management, while PaaS solutions such as Heroku or GCP App Engine have shown to facilitate deployments with pre-configured environments, prioritizing rapid development but potentially limiting customization options and control over underlying infrastructure. Continuous monitoring and proactive maintenance post-deployment are essential for sustaining application health and reliability. In essence, navigating the deployment landscape requires a blend of technical expertise, strategic planning, and adaptability to the nuances of each cloud platform.

6.1.3 IaaS or PaaS?

The fundamental question revolves around whether a business or an organization desires, and possesses the expertise and time, to handle server management in IaaS, or only focus on application developement and service while using PaaS. In a direct comparison based on equivalent parameters, IaaS typically appears cheaper, as it involves the basic infrastructure without added services. However, PaaS offers added value by eliminating server management, though both still rely on underlying infrastructure.
From a business perspective, opting for IaaS becomes more cost-effective when there is sufficient time, expertise, and resources available for proper administration, security, and monitoring of virtual servers at the OS level. This approach offers greater flexibility and control over the infrastructure, making it suitable for businesses with specific requirements or complex scalability needs. Additionally, IaaS may be preferred when there is a need for customization or integration with existing infrastructure components. However, if time is scarce for proper administration, securing, and monitoring of virtual servers at the OS level, opting for PaaS might prove more cost-effective. This approach allows for singular focus on application development without the burden of infrastructure management.

6.2 Understanding Cloud Complexity

As cloud technologies mature, the complexity of cloud environments exceeds the promises of simplicity made during the initial sales stages. If cloud computing is to become the ultimate destination for technology infrastructure, addressing this complexity becomes important sooner rather than later.
As articulated by Adrian Bridgwater in Forbes, ”Cloud is complicated because computing itself is inherently complicated, and the popularization of the cloud model approach has been constitutionally riddled with chaotic platform-level

mismatches that have slammed together with incongruent tectonic force" [38]. This complexity is compounded by the vast array of native and third-party services available, which can lead to architectures and applications that are unnecessarily convoluted.
The multitude of services offered by cloud providers, along with the diverse marketplace of third-party solutions, offers numerous options for architects and developers. However, without careful consideration, the selection and configuration of these technologies can inadvertently introduce complexity into the final solution.
Furthermore, many companies are not aware of the downsides of architectural complexity. Unlike in the earlier days of mainframe computing, where limits kept things simpler, today's cloud environment seems to offer endless options. This often leads to decisions that aren't as good during solution development. Proactively managing complexity during the initial stages of architecture and application design is crucial. However, in many cases, complexity must be tackled reactively, necessitating efforts to fix underoptimized architectural decisions made during development [39].

6.2.1 Cloud Budget Management

Effective cloud budget management has become a critical concern for organizations striving to maintain financial stability while leveraging the benefits of cloud computing. According to Flexera's 2024 report, which provides valuable insights into cloud expenditure trends, organizations are struggling with significant waste in their cloud spending.
The report highlights that, on average, organizations are wasting approximately 35% of their total cloud budgets [40]. This wastage represents a substantial loss of financial resources, underscoring the pressing need for improved cost optimization strategies within cloud environments. Despite the growing awareness of the importance of sustainability initiatives, a surprising 59% of organizations prioritize cloud cost optimization over environmental considerations [40].

6.3 Future Implications

The future of cloud computing looks promising for improving IT solutions. Cloud technology is changing how businesses work by offering cost savings, productivity, and flexibility without the need to manage infrastructure. It's becoming the new standard for business operations, and its impact on various

industries will only continue to grow. At the core of this change is edge computing and real-time processing, where data isn't only stored and analyzed in distant data centers but is handled right at the network's edge, where it originates. This approach empowers industries ranging from manufacturing to healthcare with instantaneous insights, enabling faster decision-making and enhancing efficiency.
Simultaneously, Artificial Intelligence (AI) and Machine Learning (ML) emerge as the cornerstone of cloud services. Cloud providers offer specialized tools and platforms tailored for training and deploying AI models, democratizing access to cutting-edge technologies once reserved for tech giants. This sparks innovation across sectors, from predictive analytics in finance to personalized medicine in healthcare, ushering in an era of unprecedented advancements driven by data-driven insights.
Moreover, the future of cloud computing is inherently hybrid and multi-cloud. Organizations embrace a heterogeneous mix of public, private, and hybrid cloud environments, strategically leveraging each to optimize performance, compliance, and cost-effectiveness. This heterogeneity necessitates sophisticated multi-cloud management tools, enabling seamless orchestration of workloads across disparate cloud platforms.
In parallel, serverless computing emerges as a game-changer in software development. By abstracting away infrastructure management, serverless architectures enable developers to focus solely on writing code, accelerating time-to-market and enhancing agility. This paradigm shift empowers startups and enterprises alike to innovate at scale, unencumbered by the complexities of traditional IT infrastructure.
In summary, the future of cloud computing promises innovation, integration, and transformation. It's a realm where the boundaries between the physical and digital worlds blur, enabling individuals and organizations to transcend limitations and embrace possibilities.

6.4 Future work

This section outlines significant areas that were not covered in this study, highlighting potential avenues for further investigation.

6.4.1 What has been left undone?

The prototype does not address the third requirment, *i.e.,* a yearly unavailability of less than 3 minutes; this remains an open problem. ...

6.4.1.1 Cost Models

While this study has explored cost comparisons across different CSPs, a detailed analysis focusing on pay-as-you-go versus reserved instances was not conducted. Understanding these pricing models in-depth could significantly impact cost-efficiency evaluations.

6.4.1.2 Cloud Service Provider Diversity

The research was limited to the major providers AWS, Azure, GCP, and Heroku. Expanding this to include a broader range of CSPs could uncover nuances in deployment complexities and cost structures that are specific to smaller or niche providers.

6.4.1.3 Geographical Impact on Deployment

Different geographical locations for cloud services, which can influence both performance and cost due to regional pricing and data sovereignty laws, were not considered in this analysis.

6.4.1.4 Network Costs and Scaling Strategies

The impact of network costs has been overlooked in this study. Additionally, different scaling strategies such as auto-scaling, manual scaling, and scheduled scaling and their implications on cost and resource efficiency remain unexplored.

6.4.2 Next obvious things to be done

Based on the findings and the gaps identified in this study, several areas have been earmarked for future research to extend the understanding and applicability of cloud deployment models.

6.4.2.1 Comparative Cost Analysis Over Time

Future studies should perform a comparative cost analysis over time, incorporating variables such as demand spikes and scaling strategies. This would provide a more dynamic understanding of the cost implications of different cloud services.

6.4.2.2 Cost-Benefit Analysis of Service Models

Investigating the long-term cost savings and necessary operational changes when choosing PaaS over IaaS could provide insights into the trade-offs between complexity and cost-efficiency. This includes the potential need for additional staffing or expertise.

6.4.2.3 Security Comparison Between Models

A focused study on the security implications of using different service models and deployment strategies across various CSPs would help in understanding the security trade-offs and requirements.

6.4.2.4 Environmental Impact of Cloud Services

Finally, an analysis of the environmental impact of deploying applications across different cloud platforms can address growing concerns about the carbon footprint of digital operations, potentially guiding more sustainable practices in cloud computing.

6.5 Reflections

This section explores the broader implications of the research conducted on cloud computing deployment models, particularly in terms of their economic, social, environmental, and ethical dimensions.

6.5.1 Economic Implications

The economic impact of choosing between IaaS and PaaS models is profound. Initially, IaaS may appear to be more cost-effective due to its lower starting prices and more granular control over resources, which can lead to more optimized costs in hands-on management scenarios. However, PaaS can potentially offer better cost savings in the long term through higher abstraction, reduced need for in-house expertise, and lower operational overheads. Deciding which model to adopt can influence not just direct costs but also affect strategic business decisions regarding resource allocation, scalability, and time-to-market.

6.5.2 Social Implications

From a social perspective, the selection between IaaS and PaaS has implications for workforce dynamics within the technology sector. While IaaS may require a technically skilled workforce to manage and optimize the infrastructure, PaaS simplifies many of the operational tasks, potentially shifting the focus towards more creative and developmental roles. This shift can lead to changes in job descriptions, required skills, and training needs within organizations, affecting the IT job market and education sectors.

6.5.3 Environmental Implications

The environmental footprint of deploying cloud services via IaaS or PaaS is significantly influenced by how these services are optimized and where they are located. Optimizing the configuration in IaaS setups or utilizing the inherently efficient resource management of PaaS can lead to reduced energy consumption, thereby lessening environmental impact.
Moreover, the choice of data center location also plays a critical role. Data centers in countries that prioritize low CO2 emissions and utilize renewable energy sources help in significantly reducing the carbon footprint associated with cloud computing. Selecting providers that use these green data centers can be a crucial step in aligning with sustainable practices in cloud deployments.

6.5.4 Ethical Implications

Ethically, the deployment of IaaS versus PaaS models touches on issues of data security and privacy. While both models necessitate rigorous security measures, the responsibility on the user is higher in IaaS for ensuring security configurations, compared to PaaS, where many aspects of security management are abstracted away by the provider. This shift in responsibility can raise ethical questions about the extent to which providers should be transparent about their security practices and how much control users should have over their data security.

References

[1] M. Haynie, "Enterprise cloud services: Deriving business value from cloud computing," Microfocus, Tech. Rep., 2009. [Page 1.]

[2] L. S. Vailshery, "Spending on cloud and data centers 2009-2022, by segment," August 9 2023, published by Statista. [Online]. Available: https://www.statista.com/statistics/1114926/enterprise-spending-cloud-and-data-centers/ [Page 1.]

[3] S. Srivastava, "Iaas vs. paas: How can businesses choose the best model?" Appinventiv, Tech. Rep., December 1 2022. [Online]. Available: https://appinventiv.com/blog/iaas-vs-paas/ [Page 1.]

[4] D. White, "Iaas vs. paas – understanding the difference and making the right choice," July 31 2023. [Online]. Available: https://synoptek.com/insights/it-blogs/infrastructure-performance/iaas-vs-paas-understanding-the-difference-and-making-the-right-choice/ [Page 2.]

[5] S. B. Barron, "Iaas vs. paas vs. saas: Here's what you need to know about each," July 04 2022. [Online]. Available: https://blog.hubspot.com/service/iaas-paas-saas/ [Page 2.]

[6] Z. Li, "On evaluating commercial cloud services: A systematic review," August 04 2027. [Online]. Available: https://arxiv.org/abs/1708.01412/ [Page 3.]

[7] M. K. Pratt, "Cloud computing's real-world environmental impact," June 07 2023. [Online]. Available: https://www.techtarget.com/sustainability/feature/Cloud-computings-real-world-environmental-impact/ [Page 6.]

[8] M. J. Kavis, *Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)*, January 2014. ISBN 978-1-118-61761-8 [Page 9.]

[9] OECD, "Cloud computing: The concept, impacts and the role of government policy," no. 240, 2014. [Online]. Available: https://www.oecd-ilibrary.org/content/paper/5jxzf4lcc7f5-en/ [Page 9.]

[10] G. A. L. Harrison D. Strowd, "T-check in system-of-systems technologies: Cloud computing," Appinventiv, Tech. Rep., september 2010. [Online]. Available: https://insights.sei.cmu.edu/documents/2187/2010_004_001_15167.pdf/ [Page 9.]

[11] D. G. Harkut, *Introductory Chapter: Cloud Computing*, November 05 2018. ISBN 978-1-78984-916-5 [Page 10.]

[12] T. G. Peter Mell, "The nist definition of cloud computing," september 2011. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf/ [Pages 10, 12, and 61.]

[13] "Iaas vs. paas vs. saas," August 16 2022. [Online]. Available: https://www.redhat.com/en/topics/cloud-computing/iaas-vs-paas-vs-saas/ [Pages ix and 11.]

[14] "Infrastructure as a service - worldwide," September 2023. [Online]. Available: https://www.statista.com/outlook/tmo/public-cloud/infrastructure-as-a-service/worldwide/ [Page 12.]

[15] "10 extraordinary examples of saas applications," April 05 2024. [Online]. Available: https://quixy.com/blog/examples-of-saas-applications/#10-popular-examples-of-saas-applications/ [Page 12.]

[16] R. Islam, V. Patamsetti, A. Gadhi, R. Gondu, C. Bandaru, S. Kesani, and O. Abiona, "The future of cloud computing: Benefits and challenges," *Int. J. Communications, Network and System Sciences*, vol. 16, pp. 53–65, 2023. doi: 10.4236/ijcns.2023.164004. [Online]. Available: https://doi.org/10.4236/ijcns.2023.164004/ [Page 13.]

[17] U. Hölzle, "Future of cloud computing." [Online]. Available: https://services.google.com/fh/files/misc/futurecloudcomputing.pdf/ [Page 13.]

[18] "Istio." [Online]. Available: https://istio.io/ [Page 13.]

[19] "Amazon web services." [Online]. Available: http://aws.amazon.com/ [Page 14.]

[20] "Microsoft azure." [Online]. Available: https://azure.microsoft.com/sv-se/free/cloud-services/ [Page 14.]

[21] A. Lovegood, “A beginner’s guide to cloud pricing models,” November 25 2023. [Online]. Available: https://cloudzy.com/blog/cloud-pricing-model/ [Pages 14 and 15.]

[22] C. Slingerland, “How to determine the cost of cloud computing,” December 2023. [Online]. Available: https://www.cloudzero.com/blog/cost-of-cloud-computing/ [Page 22.]

[23] “Locust.” [Online]. Available: https://locust.io/ [Pages 23 and 64.]

[24] A. S. Gillis, “cloud sla (cloud service-level agreement),” January 2024. [Online]. Available: https://www.techtarget.com/searchstorage/definition/cloud-storage-SLA/ [Page 23.]

[25] J. Heyman, L. Holmberg, A. Baldwin, and C. Byström, “What is locust?” May 19 2021. [Online]. Available: https://docs.locust.io/en/stable/what-is-locust.html/ [Page 24.]

[26] A. Richard, “Load testing using locust,” July 19 2021. [Online]. Available: https://medium.com/nerd-for-tech/load-testing-using-locust-io-f3e6e247c74e/ [Page 24.]

[27] “Minitab.” [Online]. Available: https://www.minitab.com/en-us/ [Page 28.]

[28] S. Tenny, J. M. Brannan, and G. D. Brannan, “Qualitative study.” September 18 2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK470395/ [Page 28.]

[29] Azure, “B series cpu credit model,” June 10 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/virtual-machines/b-series-cpu-credit-model/b-series-cpu-credit-model/ [Page 45.]

[30] Heroku, “Estimate the costs to run your app on heroku,” 2024. [Online]. Available: https://www.heroku.com/pricing/ [Page 46.]

[31] Atlas, “Mongodb atlas. fully managed mongodb in the cloud.” 2024. [Online]. Available: https://www.mongodb.com/products/platform/atlas-database/ [Page 48.]

[32] “Serverless vs ec2 vs containers: A comparative study,” January 16 2020. [Online]. Available: https://us.nttdata.com/en/blog/2020/january/serverless-vs-ec2-vs-containers-a-comparative-study#/ [Page 61.]

[33] I. Kosyakov, “Azure functions vs azure vms,” May 2023 2023. [Online]. Available: https://biz-excellence.com/technologies/azure-functions/azure-functions-vs-azure-vms/#/ [Page 61.]

[34] N. Agrawal, “The benefits of virtual private cloud (vpc) in aws cloud,” June 13 2023. [Online]. Available: https://medium.com/@technical_nitish/the-benefits-of-virtual-private-cloud-vpc-in-aws-cloud-a7cd024b1174/ [Page 61.]

[35] G. C. Grammatikos, “Azure virtual private network (vpn) service,” March 1 2024. [Online]. Available: https://cloudopszone.com/azure-virtual-private-network-vpn-service/ [Page 61.]

[36] “Heroku private spaces,” March 27 2024. [Online]. Available: https://devcenter.heroku.com/articles/private-spaces/ [Page 61.]

[37] J. Lindner, “Must-know cloud performance metrics,” December 19 2023. [Online]. Available: https://gitnux.org/cloud-performance-metrics/ [Page 64.]

[38] A. Bridgwater, “Why is cloud computing so complicated?” Aug 5 2021. [Online]. Available: https://www.forbes.com/sites/adrianbridgwater/2021/08/05/why-is-cloud-computing-so-complicated/?sh=5737ad942f9c/ [Page 70.]

[39] D. Linthicum, “Cloud computing,” August 10 2021. [Online]. Available: https://www.infoworld.com/article/3628245/why-the-cloud-is-so-complicatedan-explanation-not-an-excuse.html/ [Page 70.]

[40] Flexera, “Flexera 2024 state of the cloud,” March 12 2024. [Online]. Available: https://www.flexera.com/about-us/press-center/flexera-2024-state-of-the-cloud-managing-spending-top-challenge/ [Page 70.]