Jere Korhonen

Predicting Cloud Service Costs with Machine Learning

Design a Report in Power BI and Forecasting Model Using FinOps Inform Phase Principles

Tekniikan ja Innovaatiojohtamisen akateeminen yksikkö
Master of Science in Economics and Business Administration
Information systems

Vaasa 2025

UNIVERSITY OF VAASA
Tekniikan ja Innovaatiojohtamisen akateeminen yksikkö
Author: Jere Korhonen
Title of the thesis: Predicting Cloud Service Costs with Machine Learning : Design a Report in Power BI and Forecasting Model Using FinOps Inform Phase Principles
Degree: Master of Science in Economics and Business Administration
Discipline: Information Systems
Supervisor: Duong Dang
Year: 2025 **Pages:** 72

ABSTRACT:

Many companies have been shifting from their own on-premise servers to cloud services when it comes to hosting data. As data volume has increased, cloud services offer an easier way to store rapidly increasing data volume without the need to upgrade their own on-premise servers in addition to other SaaS products. As a result of fast cloud service adoption, many users and companies spend more money on the cloud than they should. Monitoring these cloud service costs is challenging, especially when working in a multi-cloud environment. The goal of this design science research thesis is to design a Power BI report for cloud service costs and predict the costs using Prophet and TimeGPT models.

This thesis is design science research. The research begins with the literature review about cloud services, FinOps, and machine learning. Then the methodology is described. After designing the conceptual model, the architecture of the system is described. The system is built using Power BI for reporting and Prophet and TimeGPT models for machine learning. Data is collected from the company's data sources into one data platform. The data includes 16 months of data. The modelling part utilizes dbt, and the modeling technique is Data Vault 2.0. The report with Azure and Snowflake cost dashboards is developed using Power BI.

In the conclusion, the results are presented as a working Power BI report, and the company is able to monitor their cloud service costs inside the organization's tenant with a predictive machine learning model. The Prophet and TimeGPT models were relatively accurate when measuring the prediction accuracy with MAE, RMSE, and MAPE. The most accurate machine Learning model is Prophet's time series model with default parameters in this thesis. TimeGPT model's performance increased when adding finetuning steps until 5 finetuning steps, over 10 finetuning steps resulted in worse performance and overfitting.

This study provided practical steps and recommendations on how to develop a cloud cost monitoring report with forecasting capabilities. Future research topics could include topics like examining machine learning models to more accurately predict cloud service and computing costs on service levels and exploring the capability to scale up or down current resources automatically based on machine learning predictions if the service is included in the prediction. This could be done by predicting the costs by using ARIMA model and adding commitment discounts in the prediction. In addition, optimizing resources using automation would be good research topic in the future.

KEYWORDS: Cloud service, FinOps, Machine Learning, Azure, Snowflake, Prophet, TimeGPT

VAASAN YLIOPISTO
Tekniikan ja Innovaatiojohtamisen akateeminen yksikkö
Tekijä: Jere Korhonen
Tutkielman nimi: Predicting Cloud Service Costs with Machine Learning : Design a Report in Power BI and Forecasting Model using FinOps Inform phase principles
Tutkinto: Master of Science in Economics and Business Administration
Oppiaine: Information Systems
Ohjaaja: Duong Dang
Valmistumisvuosi: 2025 **Sivumäärä:** 72

TIIVISTELMÄ:
Monet yritykset ovat tehneet siirtymää paikallisista konesaleista pilvipalveluiden käyttöön. Samalla datan määrä on noussut nopeasti ja pilvipalvelut tarjoavat helpon tavan säilöä dataa ilman että datan määrän kasvu johtaa paikallisten palvelimien päivitystarpeisiin. Pilvipalveluiden käyttöönoton myötä yritykset ja monet käyttäjät käyttävät enemmän rahaa pilvipalveluihin kuin heidän pitäisi. Pilvipalveluiden kustannusten seuraaminen on haastavaa erityisesti, jos yritykset käyttävät monia eri pilvipalveluiden tarjoajia. Tämän toimintatutkimus (DSR) tutkielman tarkoituksena on kehittää Power BI raportti monitoroimaan pilvipalveluiden kustannuksia ja ennustaa kustannuksia Prophet- ja TimeGPT-malleilla.

Tutkielma on toimintatutkimus. Ensimmäisessä kappaleessa luodaan teoreettinen perusta pilvipalveluille, FinOpsille ja koneoppimiselle. Seuraavaksi kuvataan metodologia, jonka jälkeen kehitetään konseptuaalinen malli ja arkkitehtuuri ratkaisulle. Ratkaisussa käytetään Power BI –järjestelmää raportointiin, minkä lisäksi ennustamiseen hyödynnetään Prophet- ja TimeGPT-koneoppimismalleja. Data kerättiin yrityksen eri järjestelmistä yhteiselle data-alustalle. Dataa on 16 kuukauden edestä. Datan mallinnuksessä hydynnettiin dbt:tä ja mallinnustekniikkana käytettiin Data Vault 2.0 -mallinnusta. Azure ja Snowflake kustannukset esiteltiin Power BI:llä tuotetulla raportilla.

Lopuksi tulokset kuvataan Power BI -raportilla, jonka avulla voidaan monitoroida kustannuksia ja seurata tulevaisuuden kustannusten ennustusta. Prophet- ja TimeGPT-mallit olivat melko tarkkoja, kun tarkkuutta ja tuloksia mitattiin keskimääräisen absoluuttisen virheen (MAE), neliöllisen keskiarvovirheen (RMSE) ja keskimääräisen absoluuttisen prosentuaalisen virheen (MAPE) avulla. Kaikista tuloksista tarkin malli oli Prophet-malli ilman mitään parametrejä. TimeGPT-mallin suoritustarkkuus kasvoi viidenteen hienosäätöaskeleeseen asti ja yli kymmenen askelta johti suorituskyvyn laskuun ja ylisovittamiseen.

Tämä tutkielma tuotti käytännönläheiset askeleet ja suositukset, kuinka kehittää pilvipalveluiden kustannusten ennustamiseen seurantaraportti ennustuskyvykkyydellä. Jatkotutkimusaiheita voisivat olla koneoppimismallien avulla tarkempi kustannusten ennustaminen resurssitasolla. Lisäksi jatkotutkimusaiheena voitaisi tutkia palveluiden kykyä skaalata resursseja ylöspäin ja alaspäin riippuen mallin ennustuksesta, jos resurssi on ennustuksessa. Nämä voitaisiin testata ARIMA-mallin avulla ja lisätä varausalennukset ennustukseen mukaan. Myös resurssien optimointi automaation avulla voisi olla hyvä jatkotutkimusaihe.

AVAINSANAT: Cloud service, FinOps, Machine Learning, Azure, Snowflake, Prophet, TimeGPT

Contents

1 Introduction 8
1.1 Research gap 9
1.2 Research problem and objectives 9
1.3 Introduction of the case organization 10
1.4 Structure of the thesis 10
2 Literature review 12
2.1 Cloud services 12
2.1.1 Cloud Service Models 14
2.1.2 Cloud Deployment Models 16
2.1.3 Cloud resources 19
2.2 FinOps 20
2.2.1 FinOps Framework 20
2.2.2 FinOps Inform 22
2.2.3 FinOps In Practice 23
2.3 Machine Learning 24
2.3.1 Prophet 25
2.3.2 TimeGPT 26
3 Research method (design science) 28
3.1 Systems Development Research Methodology (SDRM) 28
3.2 Data collection and analysis 30
4 Design development process and final artifact 32
4.1 Construct a conceptual framework 33
4.2 Develop a system architecture 35
4.3 Analyze and design the system 37
4.3.1 Data Collection and Ingestion 37
4.3.2 Data Modelling 38
4.3.3 Reporting and visualization in Power BI 39

4.4 Build the system 39
4.4.1 Set up Azure Infrastructure 40
4.4.2 Set up Snowflake infrastructure 43
4.4.3 Prophet 44
4.4.4 TimeGPT 46
4.4.5 Power BI 48
4.5 Observe and evaluate the system 51
5 Discussion 54
5.1 Limitations 56
5.2 Future research 56
References 58
Appendices 63
Appendix 1. Prophet Python Code 63
Appendix 2. TimeGPT Python Code 66

Figures

Figure 1. Cloud Service in Use, % of companies 13

Figure 2. The computing power for running applications as a cloud service, % of companies 14

Figure 3. Service models of cloud computing 15

Figure 4. Cloud Deployment Types 17

Figure 5. FinOps Framework by FinOps Foundation 21

Figure 6. Machine Learning dimensions 25

Figure 7. Conceptual model for cloud costs 34

Figure 8. Cloud Cost Management Architecture 36

Figure 9. Data Lake Architecture of the exported Azure costs 42

Figure 10. Prophet Model Forecasts 45

Figure 11. Prophet model and all data points 46

Figure 12. TimeGPT Forecasts finetuned 3 ang 5 steps 47

Tables

Table 1. System Development Research Methodology steps 29

Table 2. System Requirements 30

Table 3. Development steps 32

Table 4. ML Model Performances 51

Pictures

Picture 1. Export costs settings page 41

Picture 2. Main page of Power BI Report 50

Picture 3. Azure Cost Dashboard 50

Abbreviations

AI Artificial Intelligence
BI Business Intelligence
DS Design Science
ML Machine Learning
SDRM System Development Research Methodology

1 Introduction

Companies' data volume has plummeted in recent years, and at the same time, cloud computing has increased its popularity. 78% of companies uses cloud services in Finland in 2023 (Tilastokeskus, 2024). The drivers behind the increase of cloud computing have been the promise of cost-effective and resource-effective services (Maroc & Zhang, 2021). Bidgoli (2011, p. 23) added that cloud computing benefits are increased storage, mobility, and flexibility, and IT gets to concentrate on other tasks and automate software updates. Maroc and Zhang (2021) stated that the multi-tenant architecture is another driver for popularity, where many users, called tenants, share one service instance. Companies must monitor these services to validate the promise of cost and resource effectiveness and detect any anomalies in costs, particularly if there is an unexpected increase. Furthermore, cloud platforms have the potential to lower expenses while enhancing the responsiveness of information systems if they are implemented effectively (Bidgoli, 2011).

My master’s thesis is Predicting Cloud Service Costs with Machine Learning: Design a Report in Power BI and Forecasting Model using FinOps framework principles.

This thesis uses the design science method, which is popular in the field of information systems. More precisely, it uses the Systems Development Research Method (SDRM) from Design Science. My thesis develops a practical solution that integrates predictive modeling with Power BI’s reporting capabilities, enabling companies to forecast cloud service costs and monitor the costs of their cloud resources. Taking down the gap between cost prediction and monitoring, this research contributes to the field of information systems and develops organizational decision-making regarding cloud platforms.

1.1 Research gap

Despite the growing adoption of cloud services, research on cloud cost models and effective monitoring strategies remains limited. There are many research articles on how to optimize the number of virtual machines, yet the cost prediction is missing literature on major cloud service providers. Nawrocki and Smendowski (2024) suggested that cloud costs forecast models could focus on multiple resources rather than one virtual machine. Li and others (2022) investigated how enterprises could monitor cloud costs. They designed Smart Cloud Management Platform (SmartCMP) to monitor and optimize these cloud costs. They found that many companies have a limited understanding of their purchased cloud resources.

In addition, Cloud providers' price sheets are not clear and easy to understand and need to be more transparent. Cloud costs can quickly rocket if pay-as-go pricing models are not monitored, which makes long-term planning harder (Ponnusamy & Khoje, 2024). The FinOps Foundation has developed a framework that aims to maximize cloud benefits (*FinOps Framework Overview*, n.d.). Academic research has not investigated this framework as much as I think would be beneficial. Current research on the FinOps framework has been done by Mileski and Gusev (2023) and Smendowski and Nawrocki (2024). The first phase of the FinOps framework, known as Inform, involves tasks centered around identifying data sources related to cloud costs, usage, and efficiency (Storment & Fuller, 2023 p. 219). This is why more research is needed in this field.

1.2 Research problem and objectives

The case organization has recognized the need to develop a monitoring report for growing cloud costs and usage of multi-cloud services. The organization utilizes Microsoft Azure and Snowflake services and needs a report to monitor cloud costs. These costs can be monitored in these services, yet the organization needs an easier way

to analyze the costs in one place. In addition, the predictive machine learning model to predict cloud costs would give more insights for planning. The main objective is to develop a data model and report to monitor these cloud costs. Then, a machine learning model will be designed to predict cloud costs and identify the resources that are being consumed.

The thesis aims to answer the following research questions:
RQ1: How to build a monitoring report using the principles of the Inform phase of the FinOps framework to improve cloud cost visibility and forecasting?
RQ2: How accurately can Prophet and TimeGPT predict cloud service costs to support effective cost management?

1.3 Introduction of the case organization

The case organization works in the IT consulting field and employs 100 employees. The organization offers consulting services and tailored solutions to customers. These services include artificial intelligence and data solutions, Enterprise Resource Planning solutions, service management solutions, and other digital solutions. The services are consulting services, marketing and communication, and design services.

1.4 Structure of the thesis

The thesis is constructed in five parts. The first part is an introduction to the topic. In this part, I defined the research problem, objectives, and questions with background information. The second part of the thesis is the literature review, where cloud service, FinOps, and machine learning are defined. The third part of the thesis describes the methodology used to construct it. In this chapter, the data collection and analysis are described, along with how the requirements and data were collected. The fourth part of the thesis carries out the actual research process and design development. The last

chapter concludes and brings together the findings. The chapter discusses future research recommendations and limitations.

2 Literature review

This chapter includes a literature review of the main topics. The chapter discusses cloud services, FinOps, and machine learning.

2.1 Cloud services

Cloud services have significantly changed the way organizations access technology. Instead of purchasing their own hardware and software, companies now buy computing resources as services provided on demand via the internet (Thakur et al., 2022, p. 2; Lawan et al., 2020, p. 1). Microsoft Azure, Google Cloud Platform, and Amazon Web Services (AWS) are the three biggest cloud providers. These providers offer cloud computing services over the Internet (Lawan et al., 2020, p. 954). Mell and Grance (2011, p. 2), in their publication for the National Institute of Standards and Technology (NIST), define cloud computing as a model where consumers can use computing power on demand with widely accessible resources through a standard network connection. Additionally, cloud computing offers flexible and scalable use of shared resources from the resource pool with transparent resource usage monitoring. Thakur et al. (2022, p. 2) and Mell and Grance (2011, p. 2) similarly define cloud computing as the delivery of computing resources and capabilities accessible via the internet.

96% of companies with more than 100 employees use some unspecified cloud services. See Figure 1 for more details and a year-by-year comparison. Since 2020, the percentage of companies that use cloud services has increased steadily (Tilastokeskus, 2024). These cloud services in Figure 1 include all services from email and office programs to databases and cloud computing.

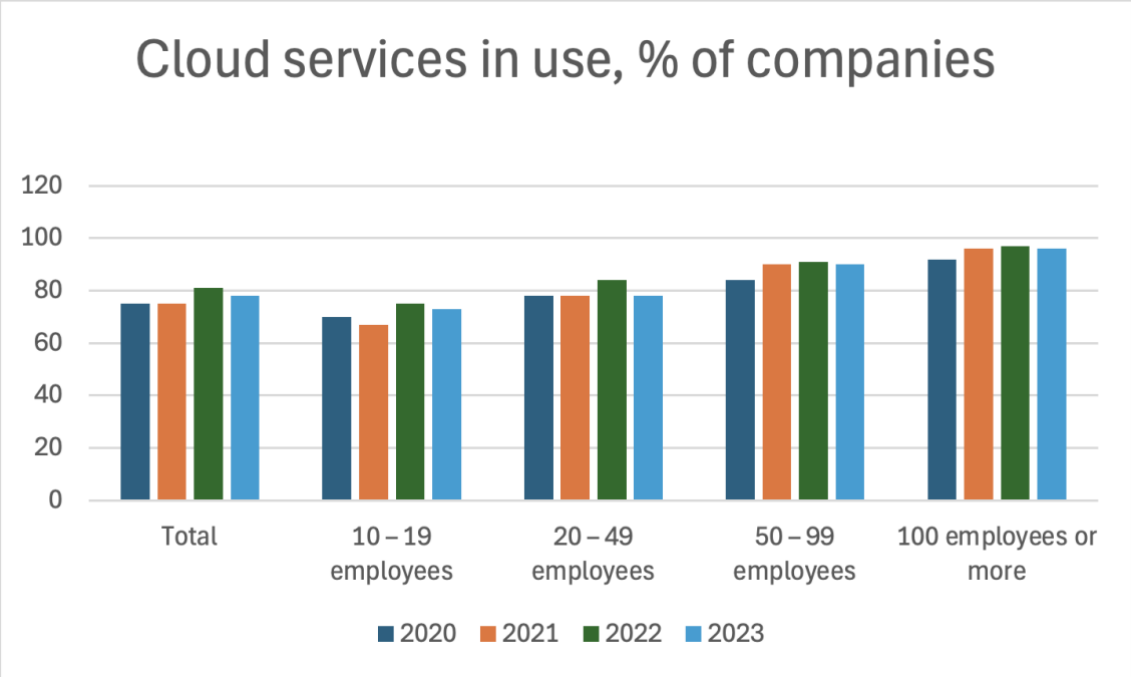


Figure 1. Cloud Service in Use, % of companies (adapted from Tilastokeskus, 2024).

The actual percentage of companies that use cloud computing power in their business is slightly lower than the number of companies that use cloud services. See Figure 2 below for statistics on computing power for running applications as a cloud service. Figures 1 and 2 show that bigger companies have invested in cloud services more than smaller companies. The investments have increased over time in all company sizes.

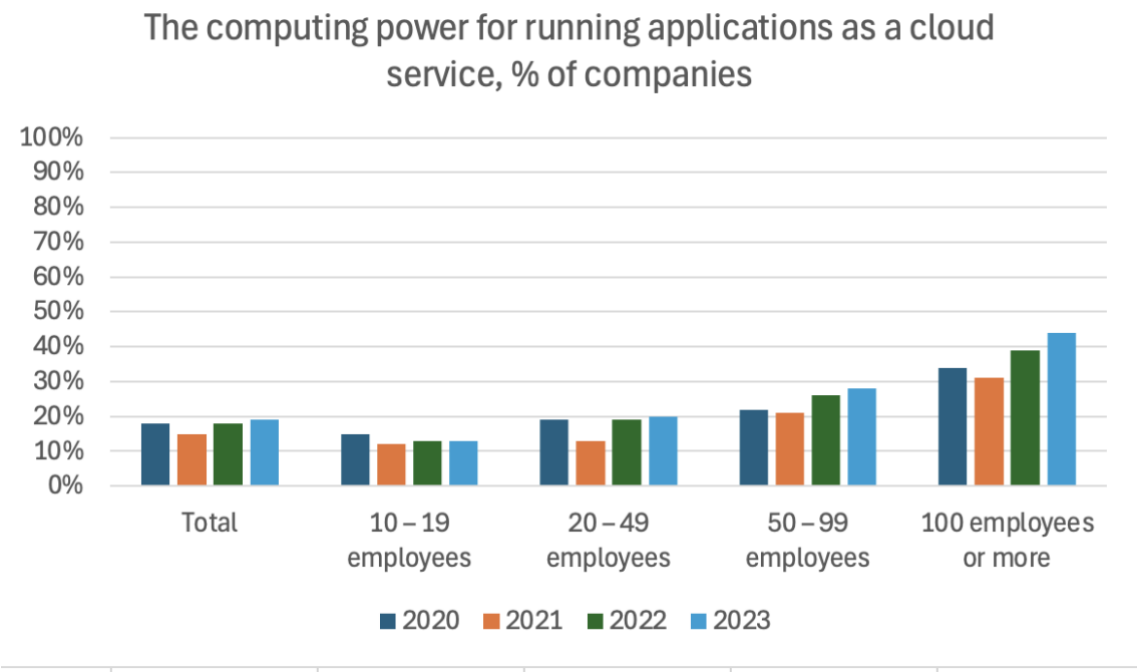


Figure 2. The computing power for running applications as a cloud service, % of companies (adapted from Tilastokeskus, 2024).

According to Mell and Grance (2011, p. 2), cloud computing services can be classified into three primary service categories—Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These categories include resources such as storage, virtual machines, and processing power.

2.1.1 Cloud Service Models

Cloud service providers offer different types of cloud computing service models. Movement from on-premises data centers and servers has rapidly progressed to cloud service models. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) are the main cloud service models (Nadeem, 2022). IaaS, PaaS, and SaaS all require different levels of knowledge to maintain. The figure 3 below describes well the differences between service models. On the figure's left is a traditional cloud service model where a company manages all the services. Moving to the right, the number of services the company manages decreases as the service provider is responsible for maintaining the services. The blue color in the figure 3 describes the

user's responsibilities and the orange color service provider's responsibilities. In a SaaS model, a user does not manage any of the services because a service provider manages all of them (M-Oliveira et al., 2023, p. 5).

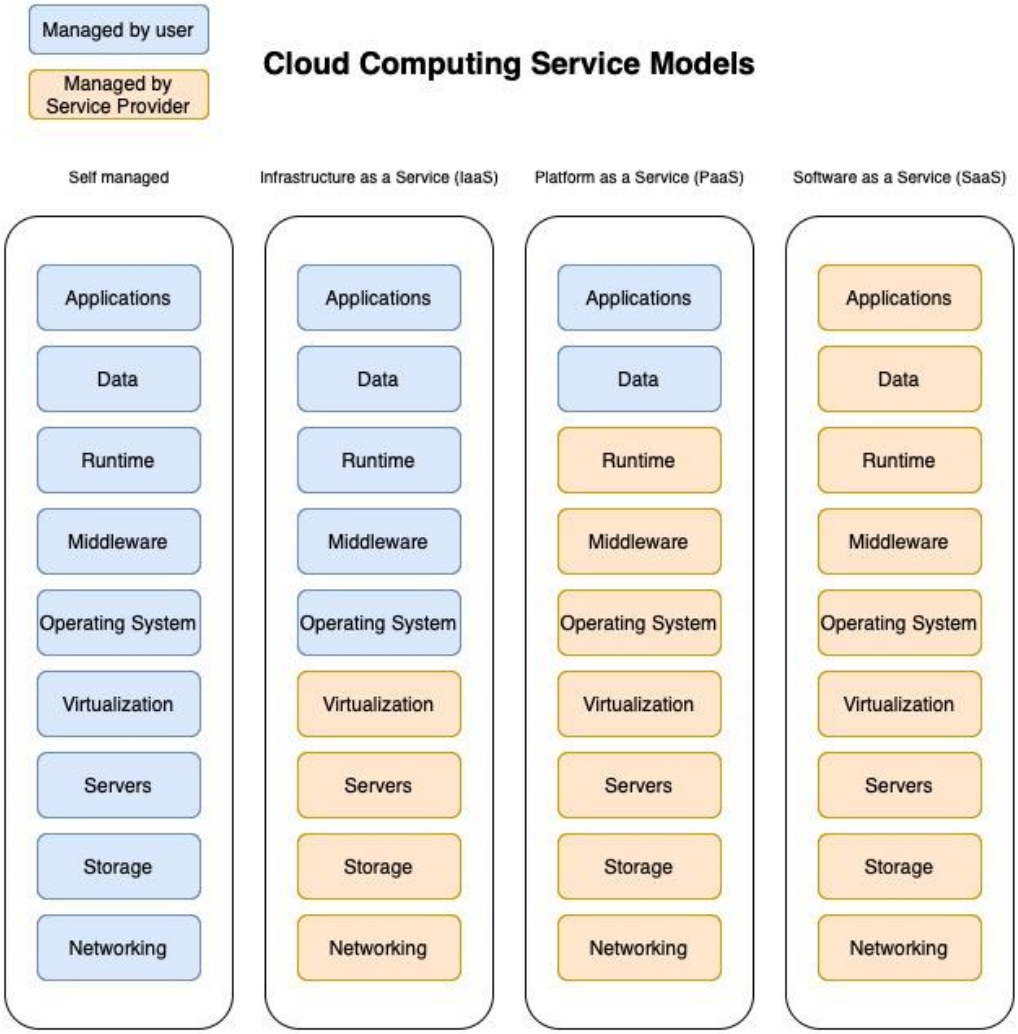


Figure 3. Service models of cloud computing (adapted from M-Oliviera and others, 2023, p. 5).

Infrastructure as a Service (IaaS) offers computing resources, storage, and networking components (Mell & Grance, 2011, p. 6). In this model, the service provider manages core infrastructure while the user retains control over operating systems, data, applications, and other software. This requires a lot of technical knowledge and resources to manage it.

Platform as a Service (PaaS) offers an environment for deploying applications (Mell & Grance, 2011, p. 6). The service provider manages the underlying infrastructure and operating systems, allowing users to focus on developing and deploying their applications. Users control only application settings and configurations, reducing the technical expertise required compared to IaaS.

Software as a Service (SaaS) is a fully managed solution where users use software applications delivered via the cloud, typically through a web page (Mell & Grance, 2011, p. 6). With SaaS, users have minimal responsibilities, limited mainly to user-specific configuration settings, and the provider manages all underlying infrastructure, including networks, servers, operating systems, and data. This service model requires the least technical knowledge to maintain. Figure 3 clearly illustrates the differences among these service models, highlighting how responsibilities shift from the company to the service provider moving from traditional models toward SaaS.

2.1.2 Cloud Deployment Models

Earlier defined service models can be implemented through various deployment models identified by Mell and Grance (2011, p. 2): private cloud, community cloud, public cloud, and hybrid cloud. This chapter explores these deployment models. According to Fatemi Moghaddam and others (2015, p. 3), cloud deployment models are illustrated in Figure 4 below. They have introduced a virtual private cloud in addition to earlier mentioned deployment models. Each deployment model is suitable for different purposes.

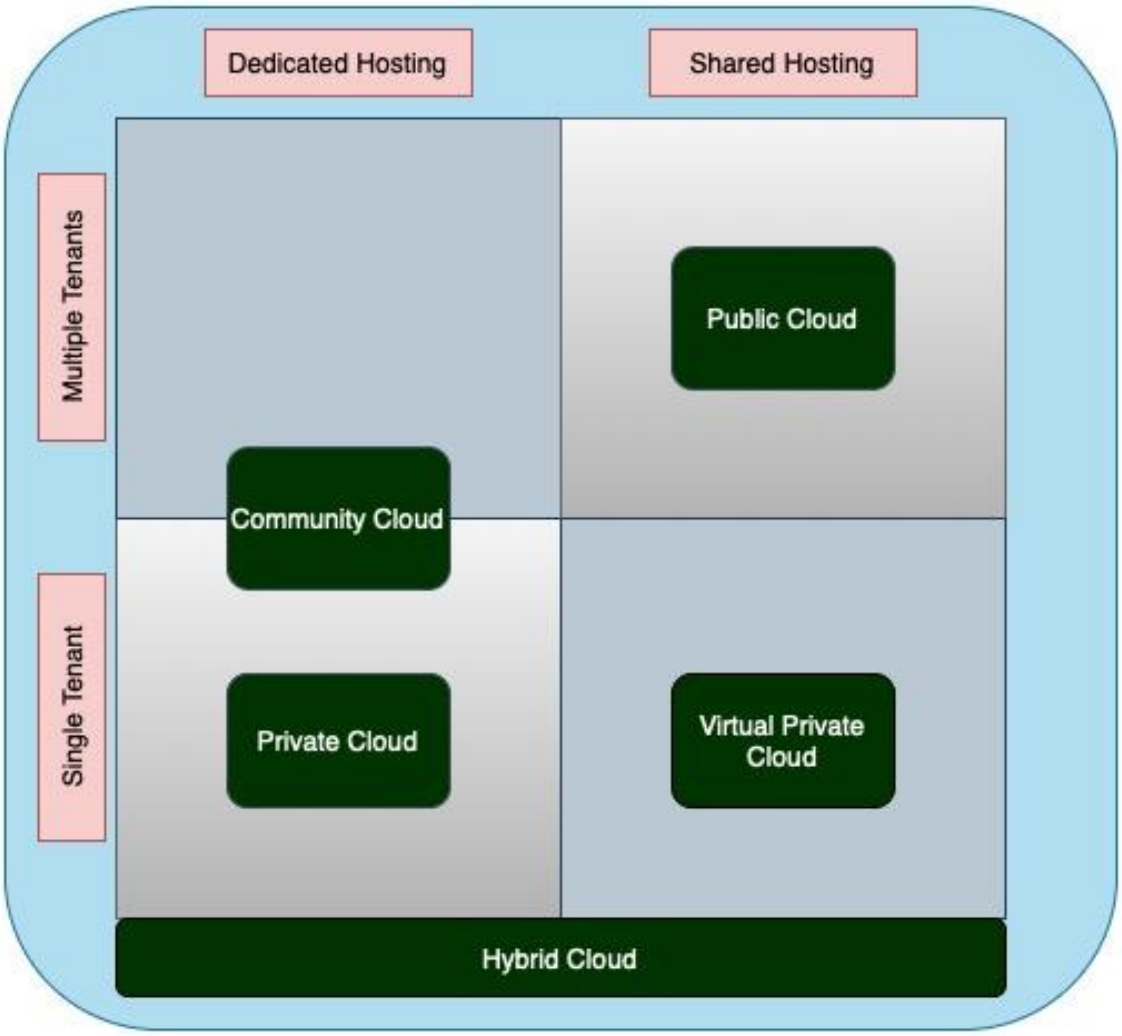


Figure 4. Cloud Deployment Types (adapted from Fatemi Moghaddam et al., 2015, p. 3).

Public clouds offer shared hosting, and multiple tenants can use it. They are usually used with secure SSL connections, for example. The public cloud infrastructure is not part of the tenant's internal infrastructure. Kilcioglu and others (2017, p. 83) state three main benefits of the public cloud. Firstly, these resources can be scaled easily. Secondly, the on-premise server's capital expenditure changes to operational costs. The third benefit is that cloud computing is a pay-as-you-go type of usage. This allows the company to always use the needed amount of computing power compared to its own on-premise servers (Kilcioglu et al., 2017). Public clouds have the potential to significantly reduce operational costs; however, a major concern is data security (Fatemi Moghaddam et al., 2015, p. 3). This type of cloud allows access to multiple users without revealing all

information to everyone. A valid example is a free email service like Gmail, provided by Alphabet, where users can access only their account after logging in (M-Oliveira et al., 2023, p. 5).

Private cloud deployment models are designed specifically for a single organization, often serving multiple internal departments or users. The organization, third-party cloud provider, or both own, manage, and operate the cloud (Mell & Grance, 2011, p. 3). Organizations select private cloud offerings when they handle sensitive data (Fatemi Moghaddam et al., 2015, p. 3). Oliveira and others (2023, p. 5) indicate that private cloud solutions are generally more costly than public cloud alternatives.

Community cloud usage is advantageous when organizations need to collaborate, share, and store their data. Oliveira and others (2023, p. 5) provided an example of a community cloud where universities collaborate on research. The community cloud can enhance this work. These community clouds can be a part of the public cloud, but this segment of the public cloud is isolated specifically for certain organizations (Fatemi Moghaddam et al., 2015, p. 4). The ownership, management, and operation of these clouds can be handled by organizations involved in community cloud. These clouds may be managed by a third-party cloud provider, or through a combination of both the provider and the participating organizations, and they can be hosted either on-premises or externally (Mell & Grance, 2011, p. 3).

A hybrid cloud combines at least two of the earlier cloud models. This model uses common or tailored technology to let data and applications move easily between systems. Microsoft offers Azure Arc resources that can manage hybrid and multi-cloud resources. The hybrid cloud merges public and private clouds, combining their best features (M-Oliveira et al., 2023, p. 5).

2.1.3 Cloud resources

Cloud providers offer hundreds of different resources that can be utilized over the Internet. The most common resources include virtual machines (VMs), storage, databases, and networking resources. These resources can be divided into three main categories: storage, compute, and network resources (Ponnusamy & Khoje, 2024, p. 3).

Storage resources store data in any format that the resource supports. These data types can be structured, unstructured, or semi-structured. Microsoft Azure offers, for example, Storage Account and SQL Server as storage resources (Azure Documentation, n.d.). I use a storage account in this thesis. A storage account is a storage space where Azure data objects, such as blobs, queues, files, and tables, can be stored. A storage account may also be called a Data Lake. It can store any data types regardless of the structure origin or format (Agarwal, 2024). I use files stored in containers within the storage account, and the queue is needed to move the files to Snowflake. Microsoft Azure offers many different databases for different purposes. Azure offering has basic SQL Server databases to its own implementation of open-source PostgreSQL engine. In addition, they offer NoSQL databases like Cosmos DB. The database offering is very wide on Azure.

Compute resources include virtual machines, containers, and serverless functions like Azure functions. Azure offers hosting, development, and deployments on its platform (Agarwal, 2024). These compute resources are used on demand, so customers may use compute whenever computing power is needed (Andersson, 2024). Compute resources are offered as pay-as-you-go or with commitment discounts. They are easy to scale up or down if usage requires computing power. Compared to old on-premises servers, compute resources are more flexible.

Networking resources allow different resources to communicate and move data between resources and geographies (Ponnusamy & Khoje, 2024). The data movement between different locations costs inbound and outbound fees. These fees are based on the amount of data volume based on GB in Azure (Microsoft Azure, n.d.). Networking in

Azure allows companies to manage scalable networking resources and allows connectivity between on-premises and public clouds (Andersson, 2024). These networking resources can manage network traffic and protect against network attacks. In Azure these networking resources may be used to make secure connection to internal resources. All of these networking resource may be monitored in Azure (Andersson, 2024).

2.2 FinOps

FinOps is a relatively new term in cloud computing. FinOps is a shortened version of Financial Operations; however, FinOps is the common term. The FinOps Foundation, part of The Linux Foundation, helps people improve their FinOps skills by building community connections, offering training and education, and sharing best practices (The FinOps Foundation, n.d.). FinOps Foundation has developed and continues to update the FinOps Framework that provides organizations with tools to succeed in managing cloud costs. See Figure 5 of the framework below. FinOps aims to optimally manage cloud services and resources to create value using cloud services (Mileski & Gusev, 2023). The FinOps framework requires a cultural shift within the organization, as it involves more than merely cutting cloud costs. Nawrocki and Smendowski (2024, p. 1) emphasize that adopting the FinOps approach means creating a culture that supports financial management, better decisions, and efficient operations. Managing the costs needs a standardized way to monitor the costs, and new resources need to be added to monitoring quickly (Lumpp et al., 2024).

2.2.1 FinOps Framework

The FinOps Framework is developed by FinOps Foundation (FinOps Framework Overview, n.d.). They have identified three phases: inform, optimize, and operate. In this thesis, we are focusing on the inform phase because there is a clear gap in the literature. See the FinOps Framework in Figure 5 below. The FinOps framework is a comprehensive

framework that covers core personas, domains, practices, scopes, phases, and principles, while combining business and technology strategy. The framework offers something for different stakeholders to rely on and serves as a straightforward guide for everyday operations.

Figure 5. FinOps Framework by FinOps Foundation (FinOps Framework Overview, n.d.).

Principles encourage the teams to collaborate, as FinOps is a cultural change. All teams have the same goal to be more efficient. If cost spikes happen, the goal is to learn from the mistakes and not to blame any team. The focus shifts from blaming to how the incident could be avoided in the future. Every team and person needs to take the ownership of the cloud costs. Maybe the most important principle for this thesis is that FinOps reports must be on time as it does not benefit anyone if reports are months or weeks late. In addition the reports should be accessible and easy to find. (Storment & Fuller, 2023).

2.2.2 FinOps Inform

The purpose of the inform phase is to make the costs visible and assign them to the correct source. The phase allows users to see their impact on the cloud bill. From the domain perspective, understanding usage & cost belongs to this inform phase category. Main activities that happen in the inform phase are to map data to the business. This can be done via tags like cost center and environment. This requires defining the tagging strategy that needs to be followed by every user in the company. If the tagging strategy is not followed, the company will have a lot of unallocated costs (Storment & Fuller, 2023). Identifying untagged resources is important as lowering the number of untagged resources increases data quality. The tags could be used to allocate the costs across different teams (Storment & Fuller, 2023). Multiple key-value pairs can be used to determine insights that are not accessible without tags, such as combinations of environment and tier tags to differentiate costs between frontend and backend. Additionally, cost center tags help group costs to the correct team and if billing supports utilizing the tags, then cloud bill can allocate costs correctly.

Defining budget and forecast is also important as budget sets the baseline for the costs and forecast predicts the future changes and might catch coming anomalies early (Storment & Fuller, 2023). Forecasting and especially accurate forecasting were one of the challenges according to the State of FinOps survey made by FinOps Foundation. The responders reported a 10% variance in the forecast compared to the actual numbers. With companies that have enormous cloud budgets of millions, this 10% is a huge variance (Storment & Fuller, 2023).

Forecasting is a core part of FinOps, and therefore, in my thesis, I try to build a Machine Learning model to predict cloud costs. Storment and Fueller (2023) define forecasting as a prediction of future costs, which is done by considering future needs and historical cost patterns. Forecasting in general is difficult because there are many variables and teams in a play. Earlier, the on-premises servers' usage was easy to predict as there were many pre-determined fixed costs that occurred when buying the servers and the

maintenance costs. Currently cloud services provide more variables that need to be monitored and they affect the overall forecasts (Storment & Fuller, 2023). They define that advanced FinOps teams who forecast costs have 10% variance when comparing actual and forecasted costs according to the FinOps Survey (Storment & Fuller, 2023). Less advanced forecasters are around 20% variance. It is not defined whether the variance is on a yearly or a monthly level.

2.2.3 FinOps In Practice

Li and others (2022, p. 176) analyzed multi-cloud cost management and highlighted its critical role in maintaining business competitiveness and efficiency. They identified several impactful benefits provided by their platform: Their platform brought AWS, Alibaba, and Azure to one place. In this thesis I chose Azure platform because all the resources are running there already and it did not make sense to move resources to different cloud provider. In addition, automatic cost export into storage account without complex integration is a good feature in Azure. AWS or Google Cloud could have been interesting choices for this thesis as well because they support FinOps Focus datasets.

Li and other's (2022) platform displayed visually the costs and allocation of how costs were allocated. The platform had the ability to process over 2 million bills each month. For this thesis, I build a simple in-house solution for a company to start tracking the costs compared to the whole platform. Li and others (2022, p. 176) used their platform to visually show the cost analysis, and for this thesis, we use Power BI for visualization. The report could be used to define non-compliant resources without tags and identify them similarly to how Li and others had used their platform. Their platform also had optimization capabilities to optimize resources that used too much compute. Nawrocki and Smendowski (2024) also pointed out the importance of optimization, which is increasing among cloud consumers. They also suggested the future research topic to forecasts costs by focusing multiple resources over virtual machines.

As companies consume more cloud services, efficiency, and the balance between costs and development speed play major roles (Li et al., 2022, p. 176). Overall, FinOps is not only a technical implementation but it also a cultural shift to maximize the cloud usage and benefits (Storment & Fuller, 2023). FinOps shifts the focus from hard values like usage costs towards conversation and working together to achieve the benefits of the cloud. In order, to have conversation of the cloud benefits the monitoring needs to be in place and everyone needs to see the results. Everyone, from developers to business leaders, has their own role in FinOps (Storment & Fuller, 2023). FinOps changes a company's culture to view cloud computing more broadly than costs and highlights how cloud resources can create value, improve efficiency, and foster collaboration across teams.

2.3 Machine Learning

Artificial intelligence (AI) usage in everyday life has grown rapidly since the Large Language Model (LLM) releases like ChatGPT and Copilot. The recent LLM development have made a huge leap forward with models like GPT-3, demonstrating remarkable capabilities in natural language processing tasks (Kasneci et al., 2023). Currently, OpenAI has developed model GPT-5 (Introducing GPT-5, 2025). Russell and Norvig (2016, p. 16) stated that the first releases of AI were McCulloch and Pitts' 1943 work on artificial neurons that were either on or off. In this thesis, the focus is still on basic machine learning models. Machine Learning is a subtopic of Artificial Intelligence. The figure below describes AI levels according to Merilehto (2018, pp. 17 & 34) and Louridas and Ebert (2016). AI is the overarching term that covers machine learning, deep learning, and their subcategories. In the figure, only machine learning is illustrated as an example with supervised and deep learning. Unsupervised learning has been left out of the figure as these models are out of the scope.

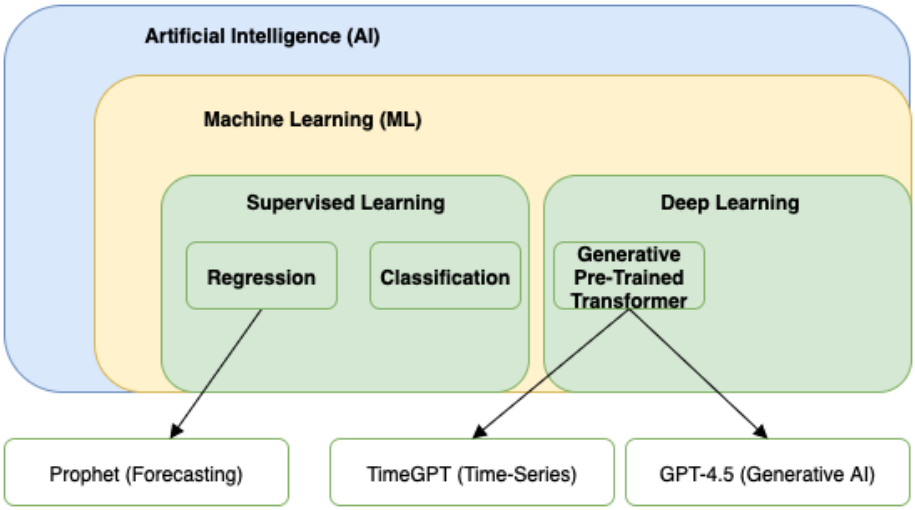


Figure 6. Machine Learning dimensions adapted from (Merilehto, 2018, pp. 17 & 34; Louridas & Ebert, 2016).

GPT stands for Generative Pre-Trained Model. Figure 6 includes the Generative AI model GPT-4.5 as an example, but it is not part of the focus of this study. The focus is on Prophet and TimeGPT models. Machine Learning can be summarized as follows: The computer is provided with a training dataset, which it analyzes to learn from. After this learning phase, the computer applies the trained model to new, unseen data (Louridas & Ebert, 2016). In this thesis, two user-friendly forecasting models, Prophet and TimeGPT, are utilized to predict cloud service costs. These models are classified as follows. Prophet is a regression-based model (Taylor & Letham, 2017). Regression models are used to predict values based on attributes. The most common regression algorithms are linear regression, decision trees, Bayesian networks, fuzzy classification, and Artificial Neural Networks (ANN) (Louridas & Ebert, 2016). TimeGPT, on the other hand, is a transformer-based model (Garza et al., 2024).

2.3.1 Prophet

Prophet is an open-source forecasting tool developed by Facebook's Data Science team (Prophet, n.d.). It was developed to analyze time-series data where seasonal variation might occur (Taylor & Letham, 2017). Prophet effectively manages gaps in data and

changes in trends and handles unusual variation well. It is available as a library for Python and R coding languages. Facebook writes that Prophet is “Accurate and fast, fully automatic, Tunable forecasts” (Prophet, n.d.). As an open-source project, Prophet is accessible to a broad user base and is simple to use.10/20/2025 10:17:00 PM

This research uses Prophet out of the box. The model runs with standard parameters because Taylor and Letham (2017) suggest that Prophet’s model often works with standard parameters. The approach allows analysts to select the appropriate components and make necessary modifications to parameters without needing to understand the model thoroughly. The prophet offers two types of models that are the saturating growth model and the default model (Daraghmeh et al., 2021). The growth model expects that the predicted value is expected to grow, but first, we need to understand how growth occurred in the past. (Taylor & Letham, 2017). In this thesis, I utilize both the logistic growth model and linear model, because when creating new resources in Azure, the costs increase in the beginning fast, but then the growth will decrease and become more stable.

There is not much research about predicting cloud service costs using Prophet. The researchers are more focused on the retail industry, where holiday times vary demand (Kumar Jha & Pande, 2021). They proved that the Prophet was able to predict retail sales quite accurately, yet they raised scalability concerns and suggested that fusion techniques could improve the accuracy. Other very popular research topics have been predicting stock markets that are classic time series datasets (du Toit et al., 2024; Saiktishna et al., 2022). Prophet could work well to predict cloud costs, as there are seasonality changes during holidays and weekends, as users are on vacation.

2.3.2 TimeGPT

Garza and others (2024) introduced TimeGPT in their paper. TimeGPT is one of the first foundation models for time series that can generate accurate predictions for data that TimeGPT has never seen or been trained on. TimeGPT is developed by Nixtla (Nixtla,

n.d.). It is a transformer-based model with self-attention. Self-attention is a way for a model to look at different data points and understand how they relate to each other. This helps the model create a better overall understanding of the entire sequence, for example, how weekends relate to timeseries data (Garza et al., 2024; Vaswani et al., 2017). Self-attention was first used in Natural Language Processing (NLP) tasks like summarization and reading comprehension (Vaswani et al., 2017).

TimeGPT was trained on a large number of public time series datasets in fields such as finance, electricity, transportation, healthcare, retail, web traffic, and economics. The datasets included over 100 billion datapoints, and the objective was to create a common global time series model that minimizes forecasting error. One of the advanced features of TimeGPT is that it can generate forecasts and detect anomaly variations out of the box without prior training data from the organization. In addition, users may fine-tune the model to adapt to the dataset's unique properties (Nixtla, n.d.).

Cloud costs prediction might vary widely between dates as usage differs from day to day; therefore, TimeGPT should be a good model to recognize these variations. Subbotin and others (2025) proved that the TimeGPT model produced better forecasts and handled time dependencies compared to classical methods. They implemented the TimeGPT model in the transportation field, which is quite different from cloud costs, but it was also a time series problem. The transformer architecture of TimeGPT has promising time series forecast results (Subbotin et al., 2025). TimeGPT model has limitations for its cost and implementation according to Subbotin and others (2025). In this thesis, I use the free trial of the TimeGPT API developed by Nixtla.

3 Research method (design science)

This chapter discusses the design science research (DSR) method and how data was collected. Design Science tries to solve or produce possible solutions to relevant business problems and is commonly used in the information systems field. March and Smith (1995, p. 253) define design science as a method that aims to create artifacts that solve defined problems. Hevner and others (2004, p. 77) emphasize that the artifact forms the foundation of design science research. DSR, as a methodology, employs a wide range of methods, depending on the type of artifact being created and the context in which it is evaluated (Hevner et al., 2004, p. 77). Venable and others (2017, p. 2) add that DSR tries to improve reality by designing new solutions. The artifacts are perishable, and the need for new or developed artifacts remains (March & Smith, 1995. p. 263).

DSR offers a variety of methodologies, allowing researchers to select the most appropriate methodology based on the specific requirements of their study. Venable and others (2017, pp. 1–10) listed six different DSR methodologies in their article that researchers can select to be used in their research:

1. Systems Development Research Methodology (SDRM)
2. DSR Process Model (DSRPM)
3. Design Science Research Methodology (DSRM)
4. Action Design Research (ADR)
5. Soft Design Science Methodology (SDSM)
6. Participatory Action Design Research (PADR)

Systems Development Research Methodology (SDRM) was identified as the most suitable approach for this study because designing the system's first artifact is quite a linear process.

3.1 Systems Development Research Methodology (SDRM)

System Development Research Methodology (SDRM) in the Design Science Research (DSR) field has existed since 1990. Nunamaker and Chen (1990, p. 639) believe that

SDRM will be a highly beneficial research methodology in the context of conducting research within the field of Information Systems. They also believe that SDRM generates valuable outcomes for information system research (Nunamaker & Chen, 1990, p. 639). Table 1 describes SDRM steps according to Venable and others (2017, pp. 2–3). The steps appear to follow a linear progression; however, the researchers are allowed to revisit and repeat any previous step of the process at any stage, as necessary (Venable et al., 2017, p. 3). Table 1 also has implementation columns that describe steps in this thesis.

Table 1. System Development Research Methodology steps adapted from (Venable et al., 2017 pp. 2–3).

Research Step	Description	Implementation
1. Construct a Conceptual Framework	Define the system requirements and understand the system building processes.	Define the scope and requirements for the system. Azure and Snowflake are cloud providers. Get the cost data from the providers. Design a conceptual data model for cloud costs to get overview of the cloud cost field.
2. Develop a System Architecture	Develop and draw an architecture design of the system.	Explain the architecture of the system.
3. Analyze & Design the System	Develop design the solution or the artefact	Build the data pipelines and infrastructure
4. Build the (Prototype) System	Learn and develop new insights through the artefact development process.	Build the actual artefact: Use the data model to build the report and train a machine learning model

5. Observe & Evaluate the System	Monitor how the developed systems is utilized and create a new models by analyzing the artifact.	Get the results of the machine learning model accuracy and critically think the report usage in company.

Chapter 4. The design development process and final artifact follow Venable and others' (2017, pp. 2-3) steps described in Table 1. The process is straightforward and linear but returns to previous steps are allowed if needed.

3.2 Data collection and analysis

In this study, stakeholders defined requirements for the system in a workshop. These requirements are used to design a service design artifact. In the workshop, the defined requirements for the system are shown in Table 2 below.

Table 2. System Requirements.

Description	Actions
Cloud Costs should update daily	Schedule the data loading
Add the cost of the Azure and Snowflake together to same report	Create a main page in Power BI where Azure and Snowflake costs are added
Resources should be identifiable	Report granularity to resource and resource group levels
Forecast Azure costs	Build a ML timeseries model using Prophet and TimeGPT

Report is visible for all the needed parties	Share the report for defined stakeholders in Power BI Service / Microsoft Fabric
Automatic data load	Make ETL Pipeline that can be scheduled

To answer Table 2 requirements, the artifact development process needs planning on how to model the data and build the architecture to support the requirements. Actual cost data was collected from the Microsoft Azure cloud service and transferred to Snowflake. Snowflake's cost data is already stored in Snowflake and updates almost in real-time. A data build tool (dbt) was used to model and transform the data for use. The dbt is a popular framework for transforming data in data workflows. The dbt supports version control, modular code structures, portability, continuous integration and continuous deployment (CI/CD), and documentation (What Is Dbt?, 2025). The data analysis will be conducted in Power BI, allowing us to analyze and visualize the cost data. The machine learning model training will be done in Snowflake Notebooks utilizing time series forecasting, Prophet, and TimeGPT.

4 Design development process and final artifact

This chapter discusses how the research was done, and how the artifact was created following to the Systems Development Research Methodology (SDRM) method of Design Science Research (DSR) (Venable et al., 2017).

The case organization has recognized the need to develop a monitoring report for multi-cloud services. The ready-made Microsoft Consumption views in Azure Cost Management are way too unmodifiable. Using these and filtering the views requires a lot of work; therefore, there is a need to have an automatically updating report that is easier and faster to use than the old and complex views. Table 3 describes the development steps.

Table 3. Development steps.

SDRM step	Steps in the study
1. Construct a Conceptual Framework	Design a conceptual data model for cloud costs to get overview of the cloud cost field.
2. Develop a System Architecture	Explain the architecture of the system.
3. Analyze & Design the System	Build the data pipelines and infrastructure
4. Build the (Prototype) System	Build the actual artefact: Use the data model to build the report and train a machine learning model
5. Observe & Evaluate the System	Get the results of the machine learning model accuracy and critically think the report usage in company.

4.1 Construct a conceptual framework

It is identified that there is not much research about cloud cost monitoring, and it is a challenge for many companies. The purpose of this thesis is to demonstrate and build a monitoring system for cloud costs with a forecasting feature. The monitoring system is developed using Azure, Snowflake, and Power BI together. This implementation tries to answer the research question 1: How to build a monitoring report using the principles of the Inform phase of the FinOps framework to improve cloud cost visibility and forecasting.

The scope of Azure and Snowflake costs was defined in the first workshop, held on 1.10.2024. During the workshop, it was agreed to develop a conceptual model of cloud costs to get an overall picture of entities. This conceptual model provides insights into what kind of data is needed for further analysis and warehouse design. Conceptual Modeling is a process where the modeler decides what to include and what to exclude in the model, according to Robinson and others (2015, p. 2816). They state that the conceptual model is not a software-specific model. It is an upper-level view of the modeled aspect. The conceptual model reduces the chances of having requirements that are incomplete or incorrect. It provides the basics for documenting actual models. It gives the model more credibility and makes it easier to verify. The model can also be reused, or part of the model can be reused (Robinson et al., 2015, p. 2819).

See the conceptual model of cloud service costs below, Figure 6. I have identified nine different entities and four different entity types. We can see the costs from the team or company perspective, which is upper-level monitoring.

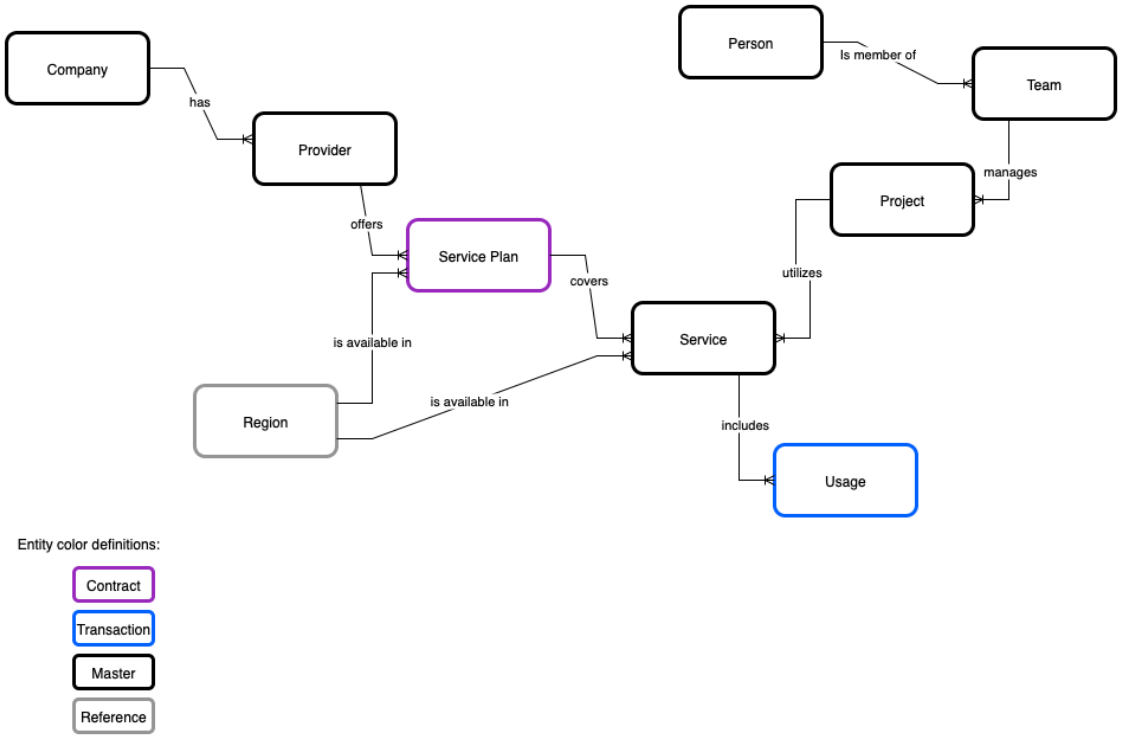


Figure 7. Conceptual model for cloud costs.

The basic entity is called the master entity, which has black borders. The master entity is a persistent and stable entity for a business. This can be easily reused in other conceptual models. The purple entity is a contract entity. A contract entity appears to be a master and a transaction entity simultaneously. It is not as persistent as a master entity but can be available for a long time. Usually, a contract has a start and end time. The duration can be short or long. Usually, the end time is not recorded anywhere, or it is not known in the beginning. The third entity type is a transaction entity, with blue borders. A transaction is an event that occurs at a specific time. This entity requires a master or a contract entity to exist. Reference entities could be marked as attributes of an entity if the modeler is not interested in them enough to make them into an entity. The region is marked as a reference entity because the location of services is an important part of cost management.

The conceptual model outlines key entities necessary for effective system design and cost analysis. The next phase involves identifying the system architecture components

needed to develop a reliable FinOps monitoring system that integrates machine learning models for future cost prediction. Once deployed, the system is designed to be user-friendly and easy to maintain. Having recognized the relevant entities, the subsequent step is to create a system architecture that facilitates the seamless flow of data and entities from the source to reporting.

4.2 Develop a system architecture

The objective was to design and develop a solution to solve cloud service cost monitoring challenges with a machine learning model to predict future costs. Current major challenges in cloud development are cost monitoring, as there are multiple data sources that need to be unified. The report needs to be up to date, and everyone needs to have access to the report. The system needs data pipelines that move data from the source to reporting, where data can be utilized. The system follows FinOps best practices and aims to make cloud cost monitoring visible for all users. There will be a reporting layer, an AI layer, a data layer, and an orchestration layer. The proposed architecture provides a clear, layered overview from data sources to the reporting layer, to facilitate efficient monitoring and daily data ingestion. Figure 8 shows all the needed resources from the upper level. The resources are very basic resources from Azure and Snowflake.

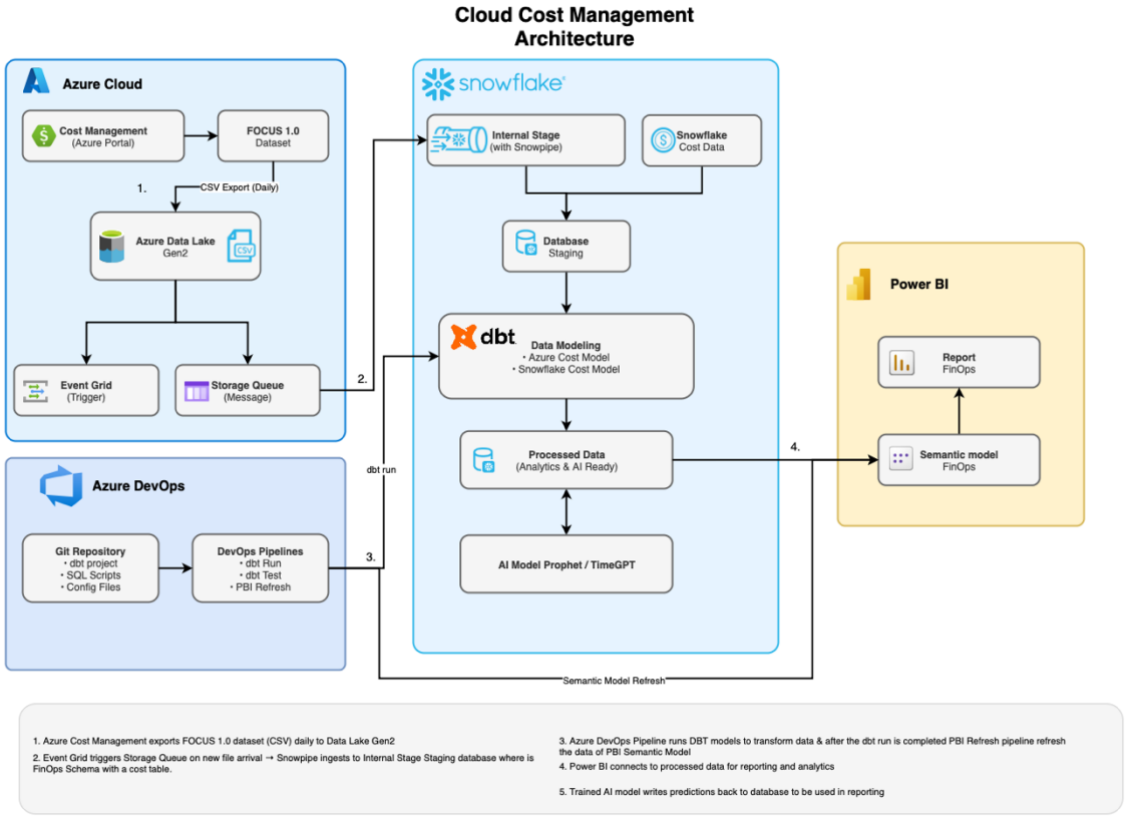


Figure 8. Cloud Cost Management Architecture.

Figure 8 shows that the solution needs many different resources. Azure Cloud has Storage Account and cost management resources with Event Grid and storage queue capabilities. Azure DevOps works as a source control GIT repository where all the code and reports are stored to support multi-developer work. There are Azure DevOps Pipelines that can be scheduled for data ingestion and Power BI refresh. These pipelines are scheduled to run every morning. On the Snowflake side, Snowpipe, AI model and databases have the most important role with dbt transformations. Snowflake is the data utilization layer. Power BI has reports and semantic models published in the Power BI Service. From Power BI Service everyone who has the access to the report is able to monitor the costs.

4.3 Analyze and design the system

The developed artifact is a system that supports cloud cost monitoring and forecasting for relevant stakeholders and their daily work, aiming to maximize the benefits of cloud development. It is not about minimizing costs but optimizing development velocity and expenses. Monitoring and forecasting are necessary to find the optimal point. This system helps identify that point. In the previous chapter the designed architecture figure 8 is used as a blueprint of the system. This approach was suggested by Nunamaker and Chen (1990, p. 635). In this chapter, I will explain the basic infrastructure that is needed for the system.

4.3.1 Data Collection and Ingestion

The Microsoft Azure Cost Management site can generate a CSV export. Data updated daily Microsoft Azure Cost Management site can generate CSV exports. Data updated daily is sufficient for our scope. Real-time data ingestion is not necessary and isn't possible, as Microsoft and Snowflake need to calculate the usage. Microsoft and Snowflake need to calculate the usage. I use batch processing, which loads the data in intervals, in this case, daily, to a system for processing (Murarka et al., 2024). This supports our scope, where real-time data analysis is not required, but daily data updates are preferred to see if some resources are overused. Snowflake stores its cost data in the Snowflake database, where we load it for our use. It is important to note that it might take up to 72 hours to get the data in Snowflake (Snowflake Documentation, n.d.). Most of the time, it is much faster than 72 hours.

Azure cost data is loaded through a storage account to the Snowflake database intra_stage, which is a staging database for raw data. The FinOps schema’s tables will be truncated every day. This approach is needed because the cost data might update a couple of days later, and we want to use the up-to-date data. We still have the history available if it is needed. We move the Snowflake cost data tables from Snowflake’s

internal Snowflake database to the intra stage as views. After this, the dbt handles the data modelling. Snowflakes' cost data movement is easier as it is already in Snowflake.

4.3.2 Data Modelling

The data that has been loaded into Snowflake needs to be modelled in order to make it useful for AI and reporting purposes. The data model part is very important, so the data is useful for end users. For data modelling, I use dbt that runs on Azure DevOps. The plan is to model data to support a star schema for the Power BI semantic model. Kimball and others (2013) describe the star schema as a structure that contains fact tables and dimension tables. Dimension tables answer questions like who, what, where, when, why, and how. The fact tables are numeric rows like the accounting values (Kimball & Ross, 2013). The actual data modelling methodology I use is Data Vault 2.0 (DV 2.0) developed by Dan Linstedt (Linstedt & Olschimke, 2016). DV 2.0 important concepts are hubs, satellites, links, and business keys. DV 2.0 aims to be as close to business as possible. It is designed in a way that modelling technique supports business following ways: integration, complexity, flexibility, and transparency (Linstedt & Olschimke, 2016). Linstedt and Olschimke (2016) clarify that DV 2.0 does not try to replace dimensional modelling that is commonly used in BI tools like Power BI.

Hubs are different business keys, like subscription, resource group, or resource, in the context of Azure. Each of these has a unique business key, for example, ID, or in the context of companies, it could be a VAT number. This ID is hashed as a hash key, and load date and record source are included in the hubs columns in order to show metadata of the record. Links are relationships between different hubs. Links make their own tables where related hubs' hash keys are stored. These links are usually many-to-many type of relationships (Linstedt & Olschimke, 2016).

Satellites complement the hub by storing descriptive data about it, such as usage value, virtual machine size, and other attributes. The hub's context can change, causing new satellites to appear or existing satellite info to be updated. These changes have been

implemented in DV 2.0, making the latest records easily accessible. This approach supports our batch processing during data loads (Linstedt & Olschimke, 2016).

4.3.3 Reporting and visualization in Power BI

Power BI is the data visualization service used to visualize the results for monitoring. Power BI needs a semantic model with measures written using the Data Analysis Expression (DAX) language. These developer written DAX measures give more control over Power BI default measures, like sum for numerical fields. With DAX, quite complex business calculations are achievable. There is no need to model and transform the data more because dbt is used in the prior step. However, we need to build measures in Power BI to make reports dynamic and establish relationships between tables. Another important part of Power BI reports is a calendar table. This calendar table is used to make reports dynamic regarding different time dimensions.

The first page of the report is a summary page where Snowflake and Azure costs are aggregated together to show the whole picture of all costs. There will be own pages for overall Snowflake and Azure costs to give more insights into each service. In addition, we have a more resource-level page for both services. The pages show more detailed consumption. All the Azure pages can be filtered by using, for example, Subscription, resource group, resource, and environment filters. These filters allow more granular views for consumption. Also, aggregation for groups is easier to manage using the earlier mentioned filters. Snowflake views could be filtered by warehouse names or databases. For the report, there is also an info page that describes the data sources.

4.4 Build the system

After developing the system architecture and identifying the needed resources, the system was built for Proof of Concept. The implementation of the system might give researchers insights into how the system could be improved or redesigned (Nunamaker

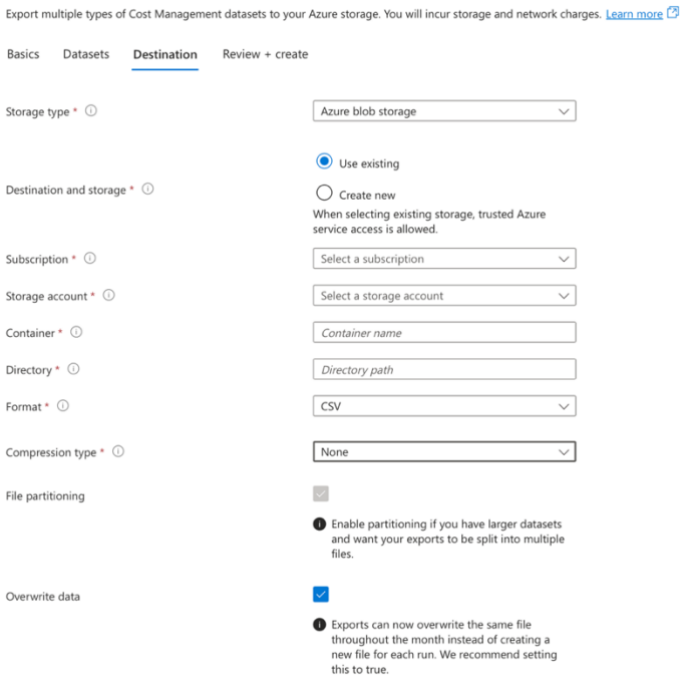
& Chen, 1990, p. 635). In the next steps, I build the system and describe the steps how the system works.

4.4.1 Set up Azure Infrastructure

This chapter explains what resources are needed on Azure side for infrastructure purposes. Azure implementation requires three essential resources: Storage Account, Azure Event Grid, and Microsoft Fabric capacity, in addition to the subscription. These resources support the data ingestion of Azure costs and reporting capabilities.

A Microsoft Azure subscription is the first thing needed in order to get started. There are different subscriptions, and I use the Azure Plan, which is the Microsoft Customer Agreement. It is important to know that not all Azure subscriptions support the FOCUS 1.0 dataset. The FOCUS dataset is required for FinOps standardized cost management reporting. The FOCUS is a standardized framework designed to harmonize cost data; it tries to ensure consistency between different cloud providers like Microsoft, Oracle, AWS, and Databricks (FOCUS™, n.d.).

Under cost management in the Azure portal, users may set up an export schedule for cost data. The storage account is needed for the export schedule. In picture 1, the desired destination and exported data format are defined. I use CSV format for simplicity without compression so I can use the data more quickly. Another option is the parquet format, which compresses the data better than CSV.



Picture 1. Export costs settings page.

I have chosen to overwrite the data because it might have been updated after the real consumption is calculated. Data duplicates are handled by dbt to model the actual data in Snowflake. The export downloads the new data every day at the same time. Usually, this is the time when the export is created. The export writes the data for the following folder structure in Figure 9 below. The archive is used for backup, and then the stage is used for the actual reporting. The stage folder will be emptied after 2 weeks.

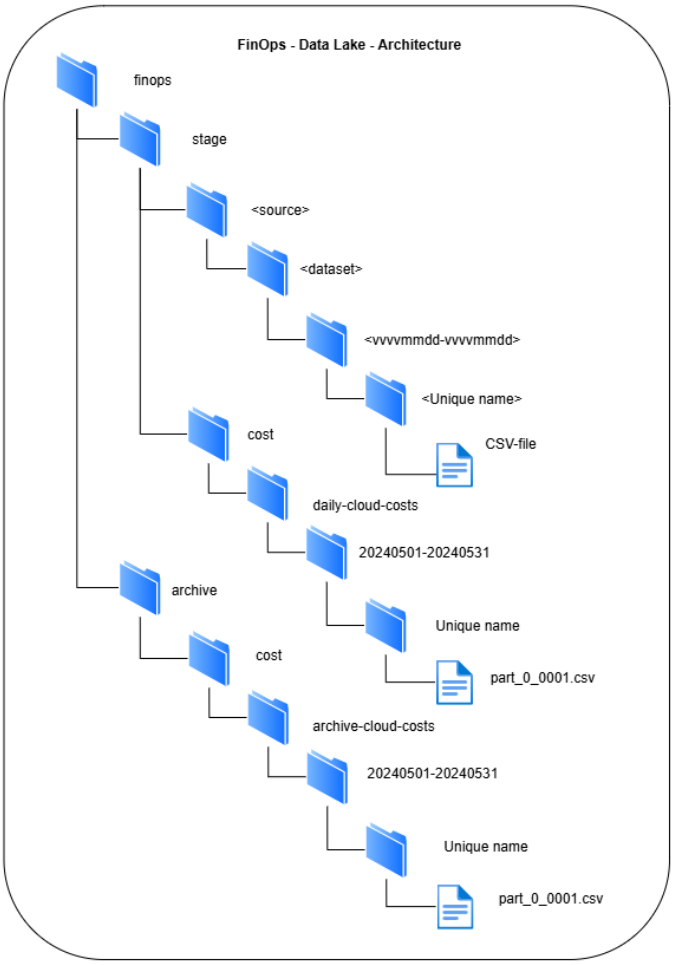


Figure 9. Data Lake Architecture of the exported Azure costs.

In Figure 9, there is a monthly folder for each month, and under the unique name, there is part_0_0001.csv, where the monthly data is stored. Each time a new export occurs, the unique name changes, overwriting the old data.

Azure Event Grid and Storage Queue play key roles in the automation part. This is an event-driven data load to Snowflake. Azure Event Grid detects new CSV files that arrive in the cost folder (see Figure 9). Storage Queue is a message sender for Snowflake's Snowpipe that loads the CSV files to Snowflake.

The last important resource is Microsoft Fabric Capacity, which is needed for the semantic model and report workloads. In Fabric, the reports can be shared across the

organization. The service name is sometimes called Power BI Service, yet currently, Fabric includes the Power BI Service features. This is a required feature because everyone who is accountable for cloud costs should see them. This is the key part of the inform phase in FinOps. More about semantic models and reports in chapter 4.4.5.

4.4.2 Set up Snowflake infrastructure

Snowflake infrastructure is designed to support data vault 2.0 data modelling methodology. There are two databases and Snowpipe for data ingestion. The first database is for staging layer, where the data is loaded to be used. A Snowpipe loads all the raw CSV files here from the Azure Storage Account to the staging layer. This staging database stores the data in its original format without any transformations. The production database uses the dbt to implement Data Vault 2.0 principles for flexibility, scalability, and historical data (Linstedt & Olschimke, 2016).

The Data Vault 2.0 schema design constructs three schemas: L00_STG (Staging layer), L10_RDV (Raw Data Vault), and L30_ID (Information Delivery). Each of these schemas plays a crucial role in Data Vault 2.0 modelling. The staging layer contains cleaned data from the staging database with record source and load timestamp generation before transformation to Data Vault methodology structures in the Raw Data Vault layer.

The Raw Data Vault layer is the core part of the Data Vault 2.0 methodology. There are core Data Vault entities: Hubs, links, and satellites. In this thesis, I do not use link tables as I make more straightforward modelling, and I am able to link dimensions with IDs to a fact table. L10_RDV schema Hubs include the business keys, load timestamps, and record sources. These are unique records. Then, satellites include the descriptive attributes of the hub. These attributes may change over time, and all the records are stored in a satellite. This approach allows selecting only the new record, or if the older records are needed, the change and history is captured. All rows from staging will be loaded to L10_RDV. Rows might include updated fields or new fields. The loaded

timestamp will differ updated rows from original rows. The decision on which rows are used in analytics will be made in the Information Delivery (ID) layer.

The Information Delivery (ID) layer is an interface for analytics and reporting. Specific business rules, data transformations, and aggregations happen in this layer. The L30_ID schema is designed for analytical workloads to support a star schema and enhance Power BI's query performance. This schema contains all the dimensions and fact tables required for cloud cost reporting. Power BI has access only to this schema to ensure data security. Only the news records from satellites are used to obtain the most current data.

4.4.3 Prophet

The Metas developed Prophet model was used with default parameters. The prophet model is suitable for time series with some seasonality, as there are available parameters to define the seasonality level (Taylor & Letham, 2017). I held the last 30 days of the data for testing and validation, and trained the Prophet model with the other 430 days.

The model was run inside the Snowflake Notebooks environment. Snowflake Notebooks support needed libraries, so there was no need to install local Python environments. Other used libraries were Pandas, Sklearn, and Matplotlib. Programming code is in Appendix 1. Prophet Python Code. Prophet model forecasts and actual values are shown in Figure 10. Below. Logistic growth predicts higher cloud costs in average.

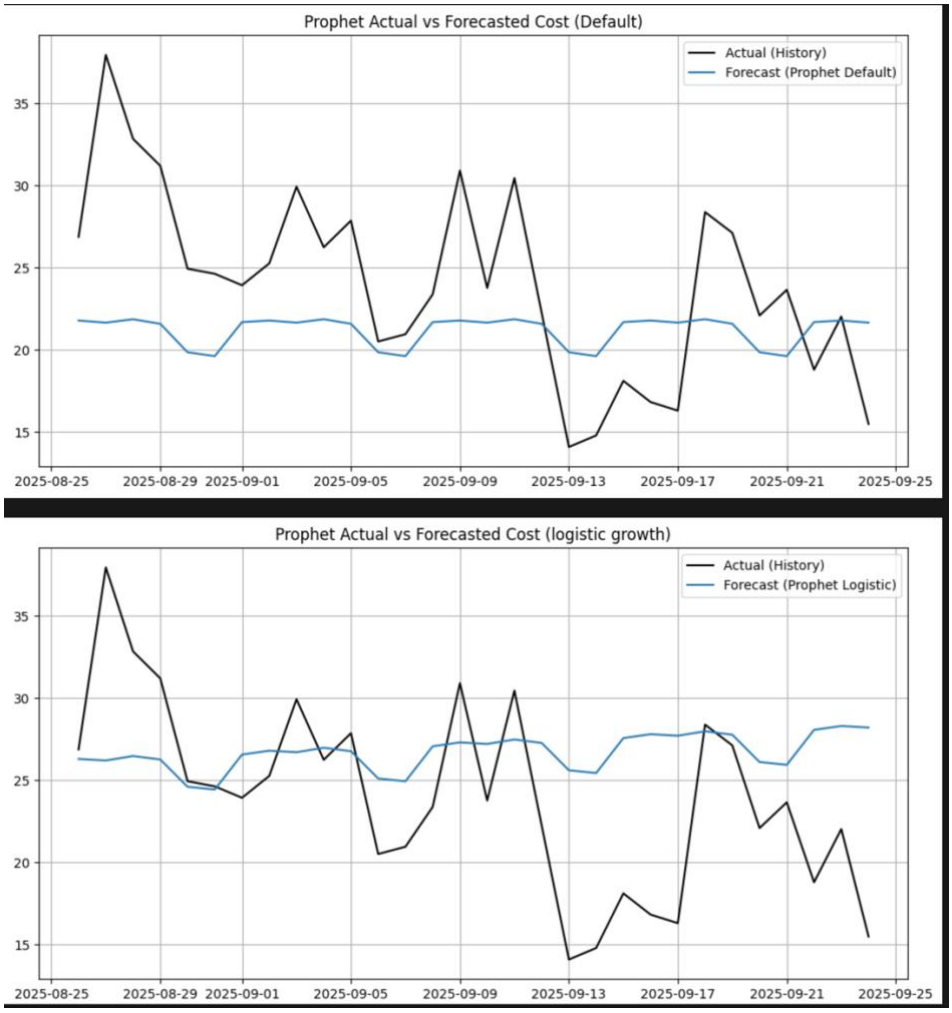


Figure 10. Prophet Model Forecasts.

Without any statistics, it is hard to say which model is more accurate from Figure 10. For that purpose, I have run three performance tests: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). These results are shown in Chapter 4.5 when evaluating the models. The Figure 11 describes all the actual data points (black dots) and the dark blue line is the forecasted value. The light blue area is the interval where Prophet estimates the most data points to be in the future.

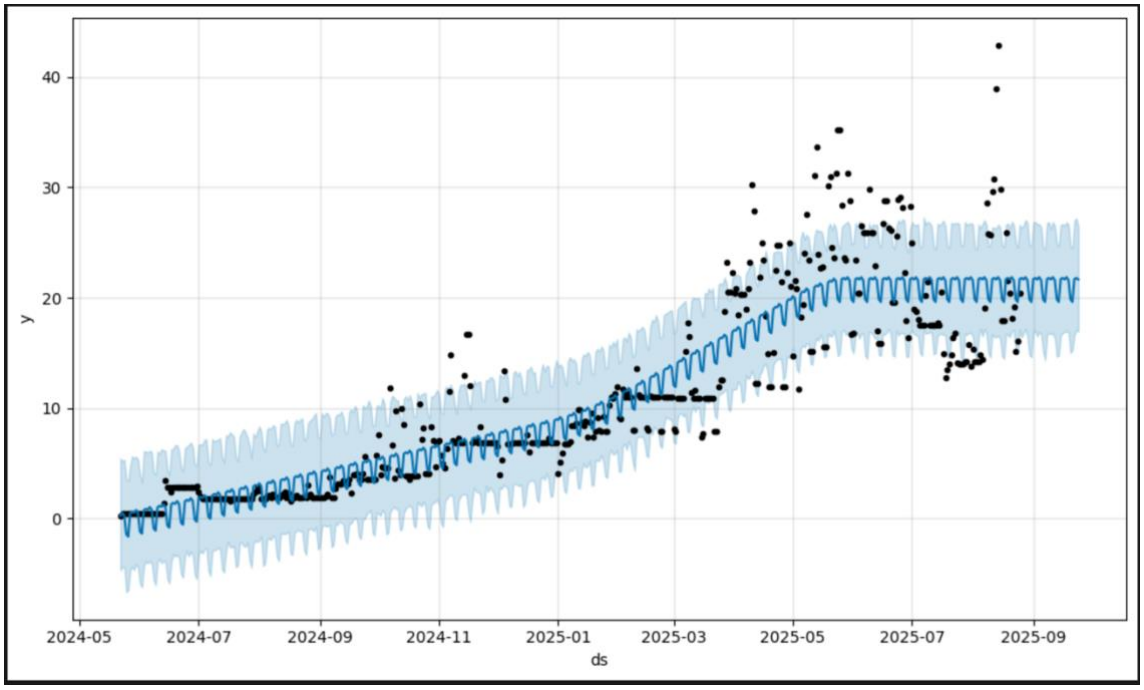


Figure 11. Prophet model and all data points.

The light blue interval captures quite well most of the data points except a couple of anomalies during May and August. In the beginning, the costs were easier to forecast. Later, forecasting is harder as data points are more scattered. This means that cloud costs have been more volatile.

4.4.4 TimeGPT

TimeGPT development was used with Nixtla's API, which was available for a 30-day free trial. TimeGPT models were run in a local Python environment. TimeGPT was also run with default parameters and fine-tuned steps 3, 5, 10, 30, and 50. For default parameters, TimeGPT uses pretrained parameter values that are trained with a huge amount of time series data (Nixtla, n.d.). It is important to recognize that adding fine-tuning steps could lead to overfitting.

TimeGPT had the same data as the Prophet models which was 491 days of Azure cost data. 461 days were used for training and the rest 30 days for testing. The default model of TimeGPT without any parameters was the baseline for TimeGPT performance. Then I

tested fine-tuning the model by adding first 10, then 30, and then 50 steps. It was proved that fewer steps performed better. Therefore, I added 5 and 3 fine-tuning steps at the end and ran the model two more times. All the performance figures are detailed in Chapter 4.5, see Table 4. When viewing Figure 12, the human eye might notice a slight difference between model finetuned 3 and finetuned 5.

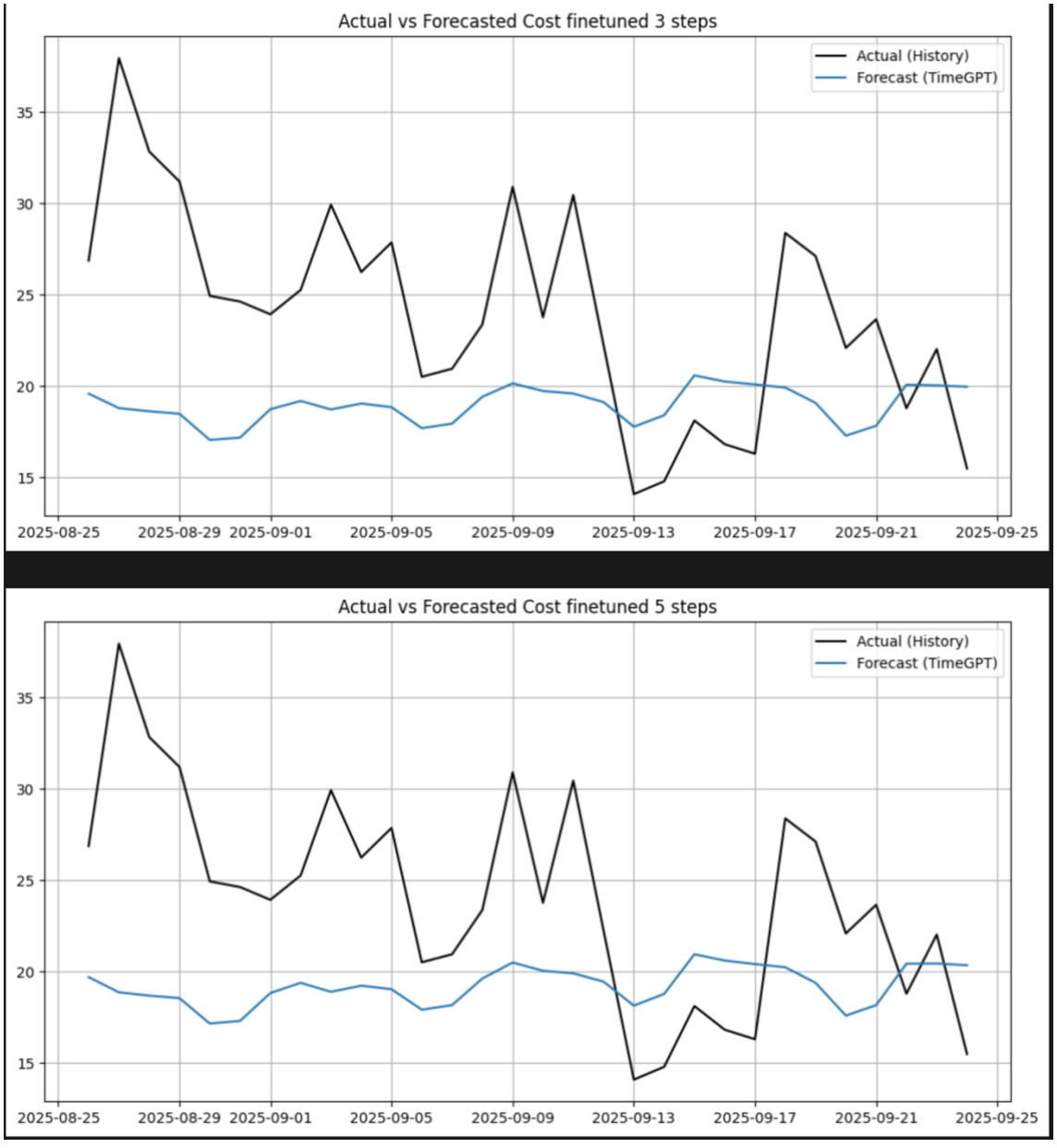


Figure 12. TimeGPT Forecasts finetuned 3 ang 5 steps.

Figure 12 illustrates the actual values left for testing and the predicted values. TimeGPT models forecast slightly under the actual line. The reason might be that the costs were smaller in the beginning see the Figure 11. The TimeGPT Python code is attached in Appendix 2.

4.4.5 Power BI

The Power BI Desktop is a development tool where report development happens. First, the data is downloaded into the Power BI Desktop. The data transformation happens after loading the data. All unused columns should be dropped from the model to improve the performance. The visualization happens after the data model with relationships is ready to use. After the development, the report is published to the Microsoft Fabric. There the report may be distributed to users in the organization. For this research I used Power BI Desktop Project file type as .pbip. This file type has enormous benefits like better source control support and folder structure as the semantic model and report are in separate folders (Microsoft, n.d.).

Key metrics for the report are total costs with different time periods and filters like environment and cost center. The report includes the following KPI metrics:

- All Costs – Includes total costs of Snowflake and Azure
- Azure Costs by resource & resource groups
- Azure Costs last year and change %
- Snowflake costs by warehouse
- Snowflake costs last year and change %
- Forecasted Azure costs

Data Analysis Expression (DAX) language is used to create KPI metric measures for reports. A simple DAX measure to sum Azure Costs utilizes the F_AZ_USAGE table and the EFFECTIVECOST column to calculate the total costs.

```
AZ Costs = SUM(F_AZ_USAGE[EFFECTIVECOST])
```

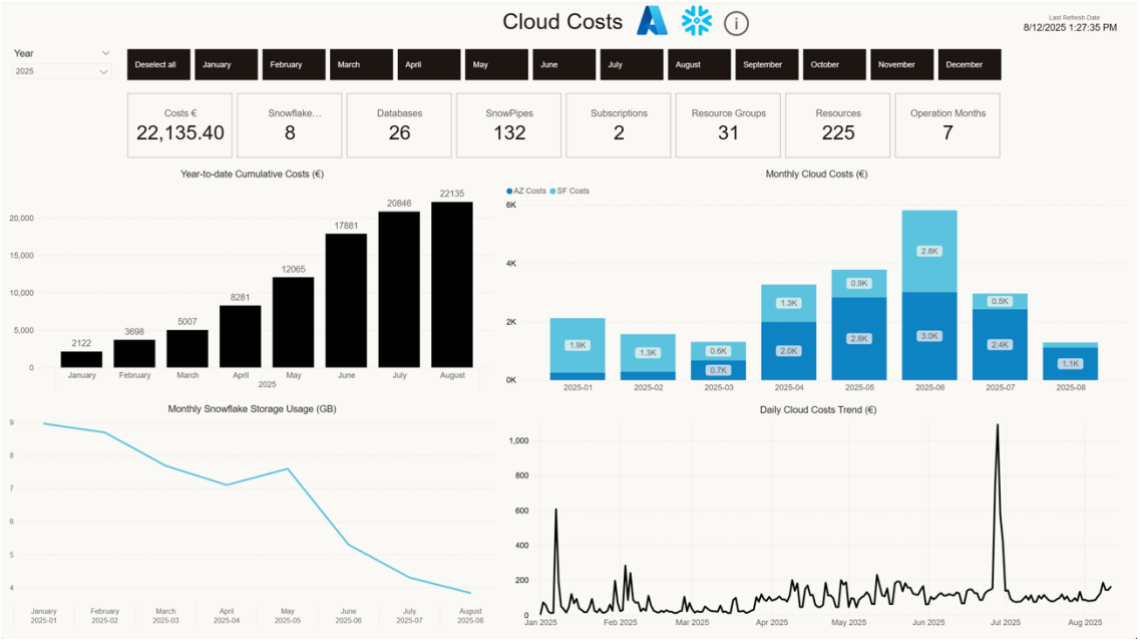
DAX code for the total costs of Snowflake and Azure shows how to use created measures inside another measure.

```
All Costs = [SF Costs] + [AZ Costs]
```

DAX measure to calculate Snowflake warehouse credits during the same time period last year. The DATEADD DAX-function moves the time window one year earlier.

```
WH Credits LY = CALCULATE(
 [WH Credits],
 DATEADD('Date'[Date],-1,YEAR)
)
```

All numerical values have been scaled by an unknown constant factor; however, the relative differences between them remain valid, allowing for direct comparison. Picture 2 shows all cloud costs combined from Azure and Snowflake. At the top of the page, there are simple KPIs. Below the KPIs, there are bar charts showing cumulative costs and monthly costs. Then, there is a legend indicating the source to compare Azure and Snowflake monthly costs in the Monthly Cloud Cost visual on the top right.



Picture 2. Main page of Power BI Report.

Picture 3. Azure Cost Dashboard.

In Picture 3, we can see the forecasted costs vs. actual costs. Forecasted costs have been slightly higher than actual costs. The page identifies the most expensive services in use. The user is able to drill down to investigate the usage more and filter the data using the

filter on top of the report. Bar charts show the costs of each month separately and cumulatively. In the next chapter, the accuracy of the ML model is evaluated using different metrics.

4.5 Observe and evaluate the system

After the first prototype is ready, the performance should be tested and test the impact on individuals or company (Nunamaker & Chen, 1990. p. 635). The monitoring system, the Power BI report, is used every day in the company, and we can declare that the built prototype was a success, and the data is updating automatically everyday by using Azure Pipelines that trigger the runs and Power BI refreshes. Automating data refreshes and loads from Snowflake and Azure Cost Management checks two system requirements.

Other system requirements included the ability to share the report across the organization. The report is shared with relevant stakeholders, and everyone has access to it. The report contains a main page with a summary of all costs. In addition, users may drill down to the costs by resource groups and service levels and see which services consume the most company resources.

The forecasting of Azure cloud costs was implemented. The ML models need a more accurate definition if the model was accurate enough. Prophet and TimeGPT were used to predict the cloud costs in Azure. The human eye could not see if the models were accurate, see the figures 10 and 12. Root mean square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) were used to test the predicted results. Table 4 shows the numbers of each model's error meters.

Table 4. ML Model Performances.

Forecasting Model	MAE	RMSE	MAPE %
TimeGPT default	6.59	7.73	25.61

TimeGPT finetune 3	6.09	7.18	24.67
TimeGPT finetune 5	5.96	7.07	24.62
TimeGPT finetune 10	6.06	7.19	24.91
TimeGPT finetune 30	6.32	7.46	25.78
TimeGPT finetune 50	6.51	7.62	26.52
Prophet default	5.10	6.14	20.86
Prophet logistic	5.01	6.40	25.11

Overall, the model performed reasonably well however, more data points in training data might enhance the accuracy even more. Timeseries data forecasting from Tesla's stock market data got MAPE% to be 21.2% (du Toit et al., 2024). MAPE% values between 20-25% are acceptable in time series forecasting and not so advanced forecasting, where volatile changes like cloud costs consumption spikes might occur. Storment and Fuller (2023) defined that advanced forecasters might get 10 percent variance, but others get 20 percent variance for forecasted values. MAE and RMSE values between 5-7 are acceptable as data point values change between 0 and 45. MAE value 5 is a small error in volatile data.

The best performing model was Prophet with default parameters when comparing error meters. For Prophet with default parameters, the MAE was 5.10 and RMSE 6.14 with MAPE% 20.86%. Prophet's logistic model had better MAE but RMS and MAPE% were higher therefore, Prophet with default parameters performed better. There were no huge differences between models. From TimeGPT, the model with 5 finetune steps performed best among all TimeGPT models. Prophet's models were both better than TimeGPT's. MAPE%, that is 20% is quite high however, it is most of the time in the right. The data contains high variance between the start date and end date values. As a

result, the earlier values create a lot of variance for the model with high anomalies when daily consumption has spiked.

5 Discussion

The purpose of this study was to develop a system that supports everyday cloud cost monitoring with prediction capabilities. This study was Design Science Research with a linear development cycle. The data was collected from different sources of company data into a common data platform from where the data was utilized in this research to develop cloud cost monitoring system in Power BI with prediction capabilities. All the components needed configuration from setting up the infrastructure to building the actual report with defined measures. This research got support from Nawrocki and Smendowski (2024) as they suggested that FinOps forecast models could focus on multiple resources than one virtual machine. The forecasting model in this thesis used summarized everyday data that included many different resources and monitoring support historical monitoring for these resources.

The developed system is now in everyday use in the case organization. The system has been helpful in gathering all the costs into one place with forecasting. Predictions performed quite well, yet the model's performance might be able to be developed more accurately. There are two main reasons for that. The first reason is the number of data points. The data collected is only 491 days, which is a small training data. The second reason is that the data collected from the beginning is not comparable to the Azure usage 1.5 years later. The usage has increased a lot.

This research had two research questions. RQ1: How to build a monitoring report using the principles of the Inform phase of the FinOps framework to improve cloud cost visibility and forecasting? The first research question tried to implement FinOps's inform phase framework to improve cloud cost visibility building the monitoring reports. It was important to take FinOps principles into account from the start because, without the principles, the tags would have been missing, and good quality data would have been missing. If the tags had been missing from the beginning and added in the middle, the tags do not move back in time in Azure, and for that time, the tags would have been empty. Because of this, in the data transformation part, there would have been more

changes to ensure quality and data manipulation. The FinOps inform phase set the requirements that the report must be accessible for everyone who uses cloud services. In this thesis, the tags clearly differentiate the environments, as some organizations might use different subscriptions for production and development environments (Soni, 2023).

RQ2: How accurately can Prophet and TimeGPT predict cloud service costs to support effective cost management? All the Prophet and TimeGPT models performed similarly and relatively well. The Prophet ML model with default parameters was a little more accurate than all TimeGPT models. The accuracy of the Prophet model with default parameters, with error metrics, was Mean Absolute Error (MAE) 5.10, Root Mean Squared Error (RMSE) 6.14, and Mean Absolute Percentage Error (MAPE) 20.86%. Error metrics show that the model performed relatively well for time series data, considering the amount of training data. TimeGPT model's accuracy increased until 5 finetuning steps, after that it started to decrease. Prophet was used to predict Tesla's stock market price with 21.2% MAPE and it was able to decrease until 12.7% using hyperparameter tuning (du Toit et al., 2024). It is important to recognize that their research had over five years of training data.

This study's research contribution was that TimeGPT and Prophet models have not been used extensively to predict cloud cost usage in the literature. This study showed real results of how the models performed with actual data from Azure usage. The model's accuracy could be increased with more data, and one could predict the usage by month, therefore, daily spikes do not appear as large in the data. Monthly predictions need more monthly data at least 24 months preferably over 36 months. Making future predictions is hard, yet when the predictions are accurate, this includes many benefits for the business. In addition, this study showed an architecture of the system that could be enabled in almost any environment.

5.1 Limitations

This research was conducted on a case company with a limited dataset in limited time, and the results shown in the figures are not for generalization. Each new case company needs to train the machine learning models independently and see the results how accurate the data is. There could be bias for usefulness of the report as the report need was recognized earlier. Other limitations are regarding the setup as the FinOps Focus schema dataset is not accessible for everyone. The schema requires actual Microsoft Azure subscription as free tiers do not support the schema. However, the some of the other Azure subscriptions support the basic cost export that could be used to ML prediction.

One more limitation is that the everyone has their own resources, and usage depends on their personal usage and these results with Prophet and TimeGPT models are not good for generalization. As du Toit and others (2024) research gave an example that their model does not take in action the relationships what caused the prediction. A limitation of this work is similar as Prophet and TimeGPT models are predictive, not causal models. They find patterns from the historical data but doesn't identify what caused the prediction.

5.2 Future research

The same research should be conducted in the future with more data points from a longer period, and the usage within the time period should be more stable from production and not in the development phase. In the development the usage increases before being more stable over time in production. It was noted that the data points were not enough to predict the future costs as there is too much variance. Other machine learning models could be used in future research like ARIMA. Adding commitment discounts to forecasting models would be interesting to see as Nawrocki and Smendowski (2024) recommended in future research. In the future, comparing actual invoices and costs from the platform could be beneficial because of the reimbursement

of usage that is platform provider’s error. These changes are not visible in historical monitoring. One key future research topic would be adding finetuned LLM’s to the analyze process and ask they to simulate different scenarios (Lumpp et al., 2024). Also according to FinOps, optimizing the resources using automation could be investigated more.

References

Agarwal, A. (2024). Ultimate Azure Data Engineering: Build Robust Data Engineering Systems on Azure with SQL, ETL, Data Modeling, and Power BI for Business Insights and Crack Azure Certifications (English Edition). Orange Education PVT Ltd. http://ebookcentral.proquest.com/lib/tritonia-ebooks/detail.action?docID=31552458

Andersson, J. C. (2024). Learning Microsoft Azure (1st ed.). O'Reilly Media, Incorporated.

Azure documentation. (n.d.). Retrieved April 26, 2025, from https://learn.microsoft.com/en-us/azure/

Bidgoli, H. (2011). Successful Introduction of Cloud Computing into your Organization: A Six-Step Conceptual Model. Journal of International Technology and Information Management, 20(1). https://doi.org/10.58729/1941-6679.1098

Daraghmeh, M., Agarwal, A., Manzano, R., & Zaman, M. (2021). Time Series Forecasting using Facebook Prophet for Cloud Resource Management. 2021 IEEE International Conference on Communications Workshops (ICC Workshops), 1–6. https://doi.org/10.1109/ICCWorkshops50388.2021.9473607

du Toit, A., Baadel, S., & Harguem, S. (2024). Predicting Tesla: Stock Market Forecasting Using Facebook's Prophet. 2024 IEEE International Conference on Artificial Intelligence and Mechatronics Systems (AIMS), 1–6. https://doi.org/10.1109/AIMS61812.2024.10513215

Fatemi Moghaddam, F., Rohani, M. B., Ahmadi, M., Khodadadi, T., & Madadipouya, K. (2015). Cloud computing: Vision, architecture and Characteristics. 2015 IEEE 6th Control and System Graduate Research Colloquium (ICSGRC), 1–6. https://doi.org/10.1109/ICSGRC.2015.7412454

FinOps Framework Overview. (n.d.). Retrieved October 23, 2024, from https://www.finops.org/framework/

FOCUS™. (n.d.). FinOps Open Cost & Usage Specification. Retrieved August 13, 2025, from https://focus.finops.org/

Garza, A., Challu, C., & Mergenthaler-Canseco, M. (2024). TimeGPT-1 (No. arXiv:2310.03589). arXiv. https://doi.org/10.48550/arXiv.2310.03589

Hevner, A., R, A., March, S., T, S., Park, Park, J., Ram, & Sudha. (2004). Design Science in Information Systems Research. Management Information Systems Quarterly, 28, 75.

Introducing GPT-5. (2025, October 6). https://openai.com/index/introducing-gpt-5/

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., … Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, 103, 102274. https://doi.org/10.1016/j.lindif.2023.102274

Kilcioglu, C., Rao, J. M., Kannan, A., & McAfee, R. P. (2017). Usage Patterns and the Economics of the Public Cloud. Proceedings of the 26th International Conference on World Wide Web, 83–91. https://doi.org/10.1145/3038912.3052707

Kimball, R. B., & Ross, M. (2013). The data warehouse toolkit: The definitive guide to dimensional modeling. Indianapolis, IN: Wiley.

Kumar Jha, B., & Pande, S. (2021). Time Series Forecasting Model for Supermarket Sales using FB-Prophet. 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 547–554. https://doi.org/10.1109/ICCMC51019.2021.9418033

Lawan, M. M., Oduoza, C. F., & Buckley, K. (2020). Proposing a conceptual model for cloud computing adoption in upstream oil & gas sector. Procedia Manufacturing, 51, 953–959. https://doi.org/10.1016/j.promfg.2020.10.134

Li, F., Wu, G., Lu, J., Jin, M., An, H., & Lin, J. (2022). SmartCMP: A Cloud Cost Optimization Governance Practice of Smart Cloud Management Platform. 2022 IEEE 7th International Conference on Smart Cloud (SmartCloud), 171–176. https://doi.org/10.1109/SmartCloud55982.2022.00034

Linstedt, D., & Olschimke, M. (2016). Building a scalable data warehouse with data vault 2.0. Morgan Kaufmann. https://doi.org/10.1016/C2014-0-02486-0

Louridas, P., & Ebert, C. (2016). Machine Learning. IEEE Software, 33(5), 110–115. https://doi.org/10.1109/MS.2016.114

Lumpp, F., Braga, D., Fummi, F., & Bombieri, N. (2024). Automating FinOps in Cloud Computing: An Integrated Solution for Efficient Data Collection with Dynamic Scraper Generation. 2024 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 79–86. https://doi.org/10.1109/CloudCom62794.2024.00025

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. Decision Support Systems, 15(4), 251–266. https://doi.org/10.1016/0167-9236(94)00041-2

Maroc, S., & Zhang, J. B. (2021). Cloud services security-driven evaluation for multiple tenants. Cluster Computing, 24(2), 1103–1121. https://doi.org/10.1007/s10586-020-03178-z

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing. NIST Special Publication 800-145, 3.

Microsoft. (n.d.). Power BI Desktop projects (PBIP)—Power BI. Retrieved August 24, 2025, from https://learn.microsoft.com/en-us/power-bi/developer/projects/projects-overview

Microsoft Azure. (n.d.). Pricing - Bandwidth. Retrieved September 2, 2025, from https://azure.microsoft.com/en-us/pricing/details/bandwidth/

Mileski, D., & Gusev, M. (2023). FinOps in Cloud-Native Near Real-Time Serverless Streaming Solutions. 2023 31st Telecommunications Forum (TELFOR), 1–4. https://doi.org/10.1109/TELFOR59449.2023.10372626

M-Oliveira, F., Rocha, A. D., Alemão, D., Freitas, N., Toshev, R., Södergård, J., Tsoniotis, N., Argyriou, C., Papacharalampopoulos, A., Stavropoulos, P., Perlo, P., & Barata, J. (2023). Cloud-Based Architecture for Production Information Exchange in European Micro-Factory Context. Applied Sciences, 13(18), 10223. https://doi.org/10.3390/app131810223

Murarka, S., Jain, A., & Singh, L. (2024). Advanced Techniques in Data Ingestion and Pipelining for Scalable Big Data Platforms: A Comprehensive Review. 2024 IEEE 4th International Conference on ICT in Business Industry & Government (ICTBIG), 1–6. https://doi.org/10.1109/ICTBIG64922.2024.10911053

Nadeem, F. (2022). Evaluating and Ranking Cloud IaaS, PaaS and SaaS Models Based on Functional and Non-Functional Key Performance Indicators. IEEE Access, 10, 63245–63257. IEEE Access. https://doi.org/10.1109/ACCESS.2022.3182688

Nixtla. (n.d.). Nixtla. Retrieved August 1, 2025, from https://nixtlaverse.nixtla.io/nixtla/docs/getting-started/introduction.html

Nunamaker, J. F., & Chen, M. (1990). Systems development in information systems research. Twenty-Third Annual Hawaii International Conference on System Sciences, iii, 631–640. https://doi.org/10.1109/HICSS.1990.205401

Ponnusamy, S., & Khoje, M. (2024). Optimizing Cloud Costs with Machine Learning: Predictive Resource Scaling Strategies. 2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT), 1–8. https://doi.org/10.1109/ICITIIT61487.2024.10580717

Prophet. (n.d.). Prophet. Retrieved July 31, 2025, from http://facebook.github.io/prophet/

Robinson, S., Arbez, G., Birta, L. G., Tolk, A., & Wagner, G. (2015). Conceptual modeling: Definition, purpose and benefits. 2015 Winter Simulation Conference (WSC), 2812–2826. https://doi.org/10.1109/WSC.2015.7408386

Russell, S., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach, Global Edition. Pearson Education Limited. http://ebookcentral.proquest.com/lib/tampere/detail.action?docID=5483443

Saiktishna, C., Sumanth, N. S. V., Rao, M. M. S., & J, T. (2022). Historical Analysis and Time Series Forecasting of Stock Market using FB Prophet. 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS), 1846–1851. https://doi.org/10.1109/ICICCS53718.2022.9788231

Snowflake Documentation. (n.d.). Exploring Overall Cost. Retrieved August 2, 2025, from https://docs.snowflake.com/en/user-guide/cost-exploring-overall

Soni, M. (2023). FinOps Handbook for Microsoft Azure: Empowering teams to optimize their Azure cloud spend with FinOps best practices (1st ed.). Packt Publishing. https://doi.org/10.0000/9781801819879

Storment, J. R., & Fuller, M. (2023). Cloud FinOps (2nd ed.). O'Reilly Media, Incorporated.

Subbotin, B. S., Smirnov, Petr. I., Karelina, E. A., Solovyov, N. V., & Silakova, V. V. (2025). Comparative Analysis of TimeGPT, Time-LLM and MSET Models and Methods for Transport Telematics. 2025 Systems of Signals Generating and Processing in the Field of on Board Communications, 1–5. https://doi.org/10.1109/IEEECONF64229.2025.10948066

Taylor, S. J., & Letham, B. (2017). Forecasting at scale (No. e3190v2). PeerJ Preprints. https://doi.org/10.7287/peerj.preprints.3190v2

Thakur, N., Singh, A., & Sangal, A. L. (2022). Cloud services selection: A systematic review and future research directions. Computer Science Review, 46, 100514. https://doi.org/10.1016/j.cosrev.2022.100514

The FinOps Foundation. (n.d.). Retrieved January 18, 2025, from https://www.finops.org/

Tilastokeskus. (2024, August 19). Tietotekniikan Käyttö Yrityksissä. https://pxdata.stat.fi/PxWeb/pxweb/fi/StatFin/StatFin__icte/statfin_icte_pxt_13vg.px/table/tableViewLayout1/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Proceedings of the 31st International Conference on Neural Information Processing Systems, 6000–6010.

Venable, J. R., Pries-Heje, J., & Baskerville, R. L. (2017). Choosing a Design Science Research Methodology.

What is dbt? | dbt Developer Hub. (2025, April 4). https://docs.getdbt.com/docs/introduction

Appendices

Appendix 1. Prophet Python Code

```
#import libraries
import pandas as pd
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_absolute_percentage_error
import matplotlib.pyplot as plt
from prophet import Prophet

#Set data path
DATA_PATH = "cost_data.csv"
# Read csv file
df = pd.read_csv(DATA_PATH, sep=',', decimal='.')
#Check the data
df.head()
#rename columns to ds and y
df = df.rename(columns={"DS": "ds", "TOTAL_COST": "y"})
#Set ds as datetime
df['ds'] = pd.to_datetime(df['ds'])
#check the data if nan or 0
df.isna().sum()
#Describe the data
df.describe()

#Split the data into train and test. We have 491 days in total.
train = df.iloc[:-30]     # everything except last 30 days
test  = df.iloc[-30:]     # last 30 days

#Prophet with default parameters
```

```
m_default = Prophet()
m_default.fit(train)

#Prophet with logistic growth
# add a cap column to the train df logistic growth needs it
train['cap'] = 50
m_logistic = Prophet(growth='logistic')
m_logistic.fit(train)

future_default = m_default.make_future_dataframe(periods=30)
future_default.tail()

#add a cap column to the future dataframe
future_logistic = m_logistic.make_future_dataframe(periods=30)
future_logistic['cap'] = 50
future_logistic.tail()

#Forecast default model & logistic model
forecast_prophet_default = m_default.predict(future_default)
forecast_prophet_default[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()

forecast_prophet_logistic = m_logistic.predict(future_logistic)
forecast_prophet_logistic[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()

#Plot forecasts
forecast_figure = m.plot(forecast_prophet_default)
forecast_figure_logistic = m.plot(forecast_prophet_logistic)

#Show Prophet trend and wwekly variation
trend_figure = m.plot_components(forecast_prophet_default)
```

```
trend_figure_logistic = m.plot_components(forecast_prophet_logistic)

#Calculate metrics
#Default model
rmse_prophet = np.sqrt(mean_squared_error(test['y'], forecast_prophet['yhat'].iloc[-30:]))
mae_prophet = mean_absolute_error(test['y'], forecast_prophet['yhat'].iloc[-30:])
MAPE_prophet = mean_absolute_percentage_error(test['y'], forecast_prophet['yhat'].iloc[-30:])

#Logistic
rmse_prophet_logistic = np.sqrt(mean_squared_error(test['y'], forecast_prophet_logistic['yhat'].iloc[-30:]))
mae_prophet_logistic = mean_absolute_error(test['y'], forecast_prophet_logistic['yhat'].iloc[-30:])
MAPE_prophet_logistic = mean_absolute_percentage_error(test['y'], forecast_prophet_logistic['yhat'].iloc[-30:])



print(f"Forecast default MAE={mae_prophet:.2f}, RMSE={rmse_prophet:.2f}, MAPE={MAPE_prophet:.4f}")
print(f"Forecast logistic MAE={mae_prophet_logistic:.2f}, RMSE={rmse_prophet_logistic:.2f}, MAPE={MAPE_prophet_logistic:.4f}")

#Plot actual vs forecasted cost using matplotlib
plt.figure(figsize=(12,6))
plt.title('Prophet Actual vs Forecasted Cost (Default)')
plt.plot(df.iloc[-30:]['ds'], df.iloc[-30:]['y'], label='Actual (History)', color='black')
```

```
plt.plot(forecast_prophet_default['ds'].iloc[-30:], forecast_prophet_default['yhat'].iloc[-30:], label='Forecast (Prophet Default)', color='tab:blue')
plt.grid(True)
plt.legend()
plt.show()

plt.figure(figsize=(12,6))
plt.title('Prophet Actual vs Forecasted Cost (logistic growth)')
plt.plot(df.iloc[-30:]['ds'], df.iloc[-30:]['y'], label='Actual (History)', color='black')
plt.plot(forecast_prophet_logistic['ds'].iloc[-30:], forecast_prophet_logistic['yhat'].iloc[-30:], label='Forecast (Prophet Logistic)', color='tab:blue')
plt.grid(True)
plt.legend()
plt.show()
```

Appendix 2. TimeGPT Python Code

```
#import libraries
import os
import pandas as pd
from dotenv import load_dotenv
from nixtla import NixtlaClient
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_absolute_percentage_error
import matplotlib.pyplot as plt

#Set data path & api key
```

```
DATA_PATH = "cost_data.csv"
api_key = os.getenv("API_KEY")
#Initialize client
client = NixtlaClient(api_key)

# Read csv file
df = pd.read_csv(DATA_PATH, sep=',', decimal='.')
#Check the data
df.head()

#rename columns to ds and y
df = df.rename(columns={"DS": "ds", "TOTAL_COST": "y"})
#Set ds as datetime
df['ds'] = pd.to_datetime(df['ds'])

#check the data if nan or 0
df.isna().sum()

#Describe the data
df.describe()

#Split the data into train and test. We have 491 days in total.
train = df.iloc[:-30]     # everything except last 30 days
test  = df.iloc[-30:]     # last 30 days

#Forecast the data
forecast_df_default = client.forecast(
df = train,
h = len(test),
freq = 'D',
```

```
time_col = 'ds',
target_col = 'y',
)

#Forecast the data finetuned
forecast_df_3 = client.forecast(
df = train,
h = len(test),
freq = 'D',
time_col = 'ds',
target_col = 'y',
finetune_steps = 3
)

#Forecast the data finetuned
forecast_df_5 = client.forecast(
df = train,
h = len(test),
freq = 'D',
time_col = 'ds',
target_col = 'y',
finetune_steps = 5
)

#Forecast the data finetuned
forecast_df_10 = client.forecast(
df = train,
h = len(test),
freq = 'D',
time_col = 'ds',
```

```
target_col = 'y',
finetune_steps = 10
)

#Forecast the data finetuned
forecast_df_30 = client.forecast(
df = train,
h = len(test),
freq = 'D',
time_col = 'ds',
target_col = 'y',
finetune_steps = 30
)

#Forecast the data finetuned
forecast_df_50 = client.forecast(
df = train,
h = len(test),
freq = 'D',
time_col = 'ds',
target_col = 'y',
finetune_steps = 50
)

#Set ds as datetime
forecast_df_default['ds'] = pd.to_datetime(forecast_df_default['ds'])
forecast_df_3['ds'] = pd.to_datetime(forecast_df_3['ds'])
forecast_df_5['ds'] = pd.to_datetime(forecast_df_5['ds'])
forecast_df_10['ds'] = pd.to_datetime(forecast_df_10['ds'])
forecast_df_30['ds'] = pd.to_datetime(forecast_df_30['ds'])
```

```
forecast_df_50['ds'] = pd.to_datetime(forecast_df_50['ds'])

#calculate the metrics
mae_default = mean_absolute_error(test['y'], forecast_df_default['TimeGPT'])
rmse_default = np.sqrt(mean_squared_error(test['y'], forecast_df_default['TimeGPT']))
MAPE_default = mean_absolute_percentage_error(test['y'], forecast_df_default['TimeGPT'])

mae_3 = mean_absolute_error(test['y'], forecast_df_3['TimeGPT'])
rmse_3 = np.sqrt(mean_squared_error(test['y'], forecast_df_3['TimeGPT']))
MAPE_3 = mean_absolute_percentage_error(test['y'], forecast_df_3['TimeGPT'])

mae_5 = mean_absolute_error(test['y'], forecast_df_5['TimeGPT'])
rmse_5 = np.sqrt(mean_squared_error(test['y'], forecast_df_5['TimeGPT']))
MAPE_5 = mean_absolute_percentage_error(test['y'], forecast_df_5['TimeGPT'])

mae_10 = mean_absolute_error(test['y'], forecast_df_10['TimeGPT'])
rmse_10 = np.sqrt(mean_squared_error(test['y'], forecast_df_10['TimeGPT']))
MAPE_10 = mean_absolute_percentage_error(test['y'], forecast_df_10['TimeGPT'])

mae_30 = mean_absolute_error(test['y'], forecast_df_30['TimeGPT'])
rmse_30 = np.sqrt(mean_squared_error(test['y'], forecast_df_30['TimeGPT']))
MAPE_30 = mean_absolute_percentage_error(test['y'], forecast_df_30['TimeGPT'])

mae_50 = mean_absolute_error(test['y'], forecast_df_50['TimeGPT'])
rmse_50 = np.sqrt(mean_squared_error(test['y'], forecast_df_50['TimeGPT']))
MAPE_50 = mean_absolute_percentage_error(test['y'], forecast_df_50['TimeGPT'])

print(f"Forecast default MAE={mae_default:.2f}, RMSE={rmse_default:.2f}, MAPE={MAPE_default:.4f}")
```

```
print(f"Forecast 3 MAE={mae_3:.2f}, RMSE={rmse_3:.2f}, MAPE={MAPE_3:.4f}")
print(f"Forecast 5 MAE={mae_5:.2f}, RMSE={rmse_5:.2f}, MAPE={MAPE_5:.4f}")
print(f"Forecast 10 MAE={mae_10:.2f}, RMSE={rmse_10:.2f}, MAPE={MAPE_10:.4f}")
print(f"Forecast 30 MAE={mae_30:.2f}, RMSE={rmse_30:.2f}, MAPE={MAPE_30:.4f}")
print(f"Forecast 50 MAE={mae_50:.2f}, RMSE={rmse_50:.2f}, MAPE={MAPE_50:.4f}")

#plot the data only the last 30 days

plt.figure(figsize=(12,6))
plt.title('Actual vs Forecasted Cost no finetuning')
plt.plot(df.iloc[-30:]['ds'], df.iloc[-30:]['y'], label='Actual (History)', color='black')
plt.plot(forecast_df_default['ds'], forecast_df_default['TimeGPT'], label='Forecast
(TimeGPT)', color='tab:blue')
plt.grid(True)
plt.legend(); plt.show()

plt.figure(figsize=(12,6))
plt.title('Actual vs Forecasted Cost finetuned 3 steps')
plt.plot(df.iloc[-30:]['ds'], df.iloc[-30:]['y'], label='Actual (History)', color='black')
plt.plot(forecast_df_3['ds'], forecast_df_3['TimeGPT'], label='Forecast (TimeGPT)',
color='tab:blue')
plt.grid(True)
plt.legend(); plt.show()

plt.figure(figsize=(12,6))
plt.title('Actual vs Forecasted Cost finetuned 5 steps')
plt.plot(df.iloc[-30:]['ds'], df.iloc[-30:]['y'], label='Actual (History)', color='black')
plt.plot(forecast_df_5['ds'], forecast_df_5['TimeGPT'], label='Forecast (TimeGPT)',
color='tab:blue')
plt.grid(True)
```

```
plt.legend(); plt.show()


plt.figure(figsize=(12,6))
plt.title('Actual vs Forecasted Cost finetuned 10 steps')
plt.plot(df.iloc[-30:]['ds'], df.iloc[-30:]['y'], label='Actual (History)', color='black')
plt.plot(forecast_df_10['ds'], forecast_df_10['TimeGPT'], label='Forecast (TimeGPT)',
color='tab:blue')
plt.grid(True)
plt.legend(); plt.show()
```