university of groningen / faculty of science and engineering

Monitoring the computational waste of cloud-based applications

Kevin Gevers
June, 2022

Master Thesis Computing Science
University of Groningen, The Netherlands

Primary supervisor
Vasilios Andrikopolis
University of Groningen

External supervisor
Remco Rakers
Deloitte - Consulting - Cloud Engineering

Second supervisor
Andrea Capiluppi
University of Groningen

In collaboration with Deloitte

Abstract

Cloud applications currently do not monitor how much of their resources are being wasted. These wasted resources cost money and consume power, unnecessarily increasing the environmental impact of running the application. This thesis aims to chart the waste in cloud applications in a way that allows developers to understand where the waste is and how to reduce it. Waste in cloud applications is the amount of allocated resources that are not being actively used. To get an understanding of how developers handle resource provisioning and monitoring in cloud applications, and how waste is managed, formal interviews with experts were conducted. The findings from the interviews were compared to the current literature and requirements for a waste monitoring system have been formulated. A formal method of monitoring waste in cloud applications is proposed for this purpose. An architectural framework that can be used to implement the waste monitoring system within the already used monitoring tools or as a new tool is also presented.

Acknowledgements

I would like to thank **Vasilios Andrikopoulos** for providing great guidance and feedback during this research project. During the regular check-ins, he not only helped with the theoretical aspects but also gave tips about the project organization and planning.

Another important person I would like to thank is **Remco Rakers** from the Cloud Engineering department at Deloitte. He was also present during the regular check-ins and provided feedback from a developer's point of view.

Together Vasilios Andrikopoulos and Remco Rakers helped me make this project what it is and I could not have done it without them. I am also very grateful for their understanding and flexibility during the difficult period I went through during this project. Their speedy and detailed feedback helped me a great deal in finalizing the thesis.

Special thanks to all the people I interviewed, this has been a key aspect of this thesis and I am very happy to have found a group of knowledgeable experts in the field of cloud engineering.

Lastly, I would like to thank the Cloud Engineering Department of **Deloitte** for their support during this research project.

Contents

1 Introduction

The concept of cloud computing is well established now and is being put to practice on a massive scale. In order to make sure these cloud applications are functioning as they should they need to be monitored. A cloud monitoring system is a tool used to manage, monitor, and evaluate cloud computing architectures, infrastructures, and services. Cloud monitoring is the process of reviewing, controlling, and managing the operational and active workflow and processes within a cloud infrastructure [1]. Cloud monitoring systems are a key tool for controlling and managing hardware and software infrastructures, but also for providing information about Key Performance Indicators (KPIs) for both platforms and applications [2]. They play a key role in the performance, capacity, and capability enhancement of the cloud. Monitoring systems also assist in improving the efficiency of the cloud in terms of its efficient resource utilization, resource assignment, performance enhancement, and billing for cloud resource usage [3].

There are a lot of cloud monitoring tools available that collect logs and metrics of the various components and resources a cloud application has. A monitoring tool can show information about the performance and costs of the various components and resources, but some also offer analytics and automated actions. The collected data is presented in dashboards, which can be used by developers and staff from various other departments of a company (e.g. the financial department). The major Cloud Service Providers (CSPs) are Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). All of them offer a native monitoring tool [4] [5] [6], but there are a lot of other options, both commercial and open source.

The cost of running cloud applications is based on the duration of use, type, and quantity of resources and services that are being used. There are four different billing models used by CSPs [7].

- *Per-use*: Resources are billed per unit of time usage. This is the most common billing model and enables on-demand access to resources at any time without upfront payments. The prices and price units vary per type of resources and CSP and can change over time.
- *Subscription*: Resources are reserved in advance for a given period of time. Upfront payment, at least partial, is required for this, but often at a discount.
- *Prepaid per-use*: This uses the same model as per-use, but the billing is performed against pre-paid credit. If the credit is exceeded either the servicing is blocked or charged using the per-use model.
- *Subscription + per-use*: This uses the subscription model, but additional resources can be added upon demand using the per-use model.

Resources like compute instances are billed based on the time they are used, regardless of the utilization of these resources. Serverless functions do not require resources to be allocated and are charged on the execution time of the function. Services offered by CSPs, like managed databases or managed containers, are billed for the resources they use. There are various other services that can be used, like support plans or analytic services, which use the billing models above as well.

Something that is not properly addressed by the currently available monitoring tools is wasted resources. A cloud application has a certain capacity for its resources and the demand for those resources will fluctuate over time depending on the load on the application. The resources are over-provisioned if the capacity is higher than the demand and under-provisioned if the capacity is lower than the demand. Figure 1 illustrates over-provisioning on the left, under-provisioning in the middle, and delayed allocation on the right. If resources are under-provisioned they are unable to process the load in real-time, which causes a delay because part of the load is processed when resources become available again [8]. To avoid this a cloud system can scale out by creating additional instances to handle the heightened load. After the additional resources helped cope with the heightened load they can be destroyed, going back to the original amount

of resources. This can be done using auto-scaling rules, if certain conditions are met the system will automatically create or destroy instances. However, creating a new instance does take time and KPIs often include keeping delays to a minimum. Because of this, a larger type of instance than necessary is often used, creating over-provisioning. Whenever there is over-provisioning there are resources allocated that are not being fully utilized and thus being partly wasted. The discrepancy between demand and the capacity for over-provisioning is not monitored closely by current monitoring tools to the best of our knowledge. This waste in provisioned resources is what this thesis will aim to chart in order to give developers the required information to reduce the waste where possible.

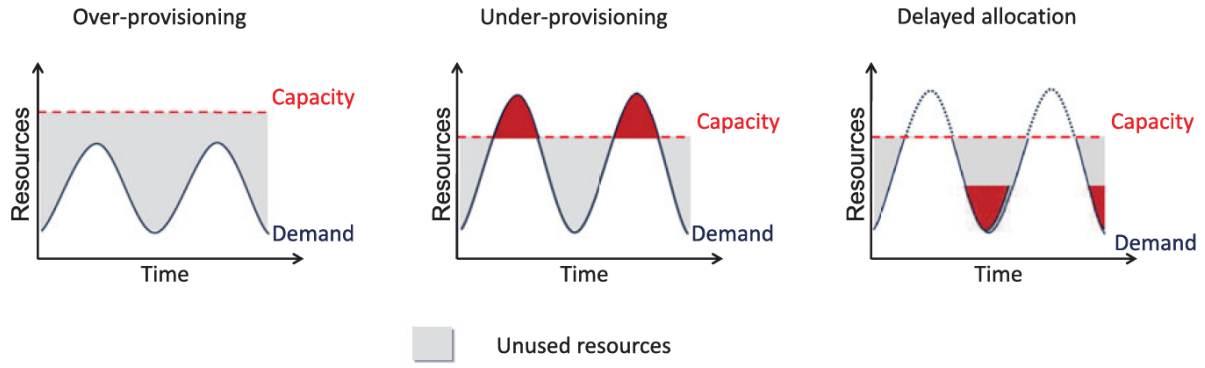


Figure 1: The cases of over-provisioning, under-provisioning, and delay caused by under-provisioning [8].

Data centers use an increasingly problematic amount of power [9], which is very costly, but also very bad for the environment. If allocated resources are not being fully utilized, smaller or fewer resources would likely suffice. Using smaller or fewer resources would reduce the cost of running a cloud application. This would also reduce the power needed for the resources of a cloud application and thus reduce the negative impact on the environment. This makes it interesting to know if and where cloud applications have waste, allowing action to be taken in reducing the waste.

Compute resources, like Virtual Machines (VMs) and containers, require the entire resource to be available regardless of their utilization. This means that it is likely these types of resources will have waste. Because VMs and container services are billed based on the duration, type, and number of resources used, reducing the waste will reduce costs. Because VMs and containers are likely to have waste they will have a focus during this research project. Serverless functions, on the other hand, do not require resources to be provisioned and are only billed for the duration of the execution of the function. This billing is done using a granularity of 1ms [10] [11], so the costs very closely match the actual usage. This means that it is unlikely that there is significant waste in serverless functions. Due to this serverless function will largely be out of scope for this research project. Managed services, like databases or storage, do require resources to be allocated and can thus contain waste. These are also billed based on the size and duration of use, thus reducing waste could reduce costs.

To achieve the goal of mapping waste to allow action to be taken in reducing the waste the main research question has been formulated as Q1. The research question can be split up into two sub-questions, namely SQ1 and SQ2. SQ1 focuses on defining and calculating waste in cloud applications and SQ2 focuses on how to present this to users of cloud monitoring tools.

Q1: *How to monitor and calculate resource waste in cloud-based applications?*

SQ1: *What is the most efficient and accurate way to compute the waste in provisioned resources for a cloud application?*

SQ2: *How can the waste be best represented to the user to get better insights?*

The thesis consists of the following chapters. Chapter 2 discusses related work, literature, and technologies. Chapter 3 outlines the interviews, including the methodology, case study design, analysis, findings, and a comparison between the literature and the findings. Chapter 4 explains the *Waste Monitoring System* and the formal definition of waste. Chapter 5 evaluates the proposed framework and presents the results of the questionnaire that was conducted to get feedback on the *Waste Monitoring System* from the interviewees. It also discusses various points of improvement based on their feedback. Finally, Chapter 6 concludes the thesis and outlines the future work.

2 Related Work

In this chapter, the related work will be discussed. First, the necessary background information about cloud monitoring will be provided. The earlier projects that this research project builds upon are discussed next, as well as related literature about estimating costs for cloud applications and monitoring architectures. Lastly, current monitoring tools and dashboards will be discussed as related technologies.

2.1 Background

A cloud monitoring system is an automated and manual tool used to manage, monitor, and evaluate cloud computing architectures, infrastructures, and services. Cloud monitoring is the process of reviewing, controlling, and managing the active processes within a cloud infrastructure [1]. In this section, the reasons for cloud monitoring and the cloud monitoring properties will be explained.

2.1.1 Reasons for Cloud Monitoring

There are various reasons for cloud monitoring, the five most important ones are discussed below.

Performance management/troubleshooting [2] [3]: Likely to be the most important reason for cloud monitoring is performance management. This allows the user to see how the application and its resources are performing and in case the performance is not what it should be the issue can be tracked down. This allows the user to identify issues and determine the cause thereof.

Resource management [1] [2] [3]: Another very important reason to monitor cloud applications is resource management. By accurately capturing the current state of the resources and the load that is put on them, the number of resources can be adjusted by adding or removing instances to match the current load. This can be done manually or with auto-scaling rules.

Billing [1] [2] [3]: The use of different services and resources incur costs and this needs to be monitored in order to know what exactly is causing these costs. CSPs can use billing models with a different granularity for different services and resources. Being able to track the costs makes it possible to try to reduce the costs by using a different configuration or using different services.

Security management [1] [2]: Security and privacy in the cloud are essential, especially considering business-critical applications or government applications. By monitoring the security of a cloud application it is ensured that any breaks in the security are found relatively early and the appropriate action can be taken.

SLA management [1] [2]: Monitoring is mandatory and instrumental in certifying Service Level Agreement (SLA) compliance when auditing activities are performed to respect regulation [2]. To comply with the legislation and other agreements, it needs to be possible to check those terms and conditions. By monitoring the cloud application this information becomes available and the compliance can be verified.

2.1.2 Cloud Monitoring Properties

A monitoring system is required to have several properties in order to operate properly. The paper by Aceto et al. [2] describes the following set of such properties.

Scalability: When a monitoring system can cope with a large number of probes it is considered scalable. A probe is a small program that has to be deployed on a resource to collect data and send it to the monitoring tool. A large number of parameters need to be monitored for a large number of resources, making this a very

important property. A scalable monitoring system should be able to efficiently collect, transfer, and analyze this large amount of data without negatively affecting the cloud application that is being monitored.

Elasticity: A monitoring system is elastic if it can cope with dynamic changes in the resources that need to be monitored, meaning that it should be able to handle the creation and destruction of (virtual) resources, which occurs when a system is scaling in or scaling out.

Adaptability: If a monitoring system is able to adapt to varying computational and network loads, without impeding the cloud application itself, it is adaptable. The monitoring tool requires network bandwidth and computational resources as well, but it is vital that it does not impact the functioning of the cloud application. This means the overhead of monitoring a cloud application should be kept to a minimum.

Accuracy: The accuracy of a monitoring system is how close the measured values are to the real values to be monitored. The accuracy is vital due to the activities that make use of the data. For example, if the data is used to identify an issue in a cloud application, inaccuracy may lead to identifying the wrong cause.

Timeliness: The monitoring system is timely if detected events are available on time for their intended use. Not having the data available on time can cause alerts not to be raised or automated rules not to take effect at the correct moment. The sampling interval of metrics needs to be low enough for the specific use case. A system that has a lot of traffic requires data to be sampled at a much higher frequency than a system with stable small amounts of traffic.

Autonomicity: Cloud infrastructures are providing on-demand self-service and rapid elasticity continuously with minimal service interruptions. It is important for a monitoring system to be able to detect and react to changes, faults, and performance degradation, without manual intervention.

Comprehensiveness, Extensibility, and Intrusiveness: A monitoring system is comprehensive if it supports different types of resources and supports several kinds of monitoring data. This is useful because it allows the user to adopt a single monitoring system, independently of what monitoring data is actually used. A monitoring system is extensible when the support can easily be extended, for example, through plug-ins or custom functions. Like comprehensiveness, this allows the adoption of a single monitoring system because missing features can be added manually without having to add an additional monitoring system. The intrusiveness is based on how much the cloud application needs to be modified for the monitoring system to be adopted. This means that a monitoring system should have as low as possible intrusiveness. These three properties are a trade-off in monitoring tools. A higher comprehensiveness will likely cause higher intrusiveness but requires less extensibility.

Resilience, Reliability, and Availability: Since monitoring a cloud application is essential for things like resource, performance, security, and billing management, this means a monitoring system needs to be resilient, reliable, and available. To be resilient, the system needs to be able to withstand component failures while continuing to operate normally. Allowing the user to justifiably trust the monitoring system. The system is reliable when it can perform a required function under stated conditions for a specified period of time. If the system can provide services according to the system design whenever the user requests them means that it is available.

2.2 Related Literature

2.2.1 Prior Works

This research project is building on the theses projects of Spina [12] and Vogel [13]. In Spina's work, the concern for the lack of a non-proprietary tool to monitor cost and waste in a generalized manner on cloud deployments in the Virtual Machine as a Service (VMaaS) delivery model is addressed. A design for a

monitoring tool that uses probes to push data to a service was presented. The collected data is then stored in a database. Accessing this data with an API was left as future work. The work focuses on cost optimization but does consider resource waste as well.

Vogel's work was the successor to this and has a larger focus on resource waste. This project focuses on the costs of running containers in the cloud. A pricing model is presented, along with Formula 1 for calculating the effective costs and Formula 2 for calculating the wasted costs [13]. A cloud monitoring solution is presented that allows for cost optimization on a deployment level, which is verified by deploying it on an industrial cloud system. This system has good results in monitoring costs and waste, but is limited to using Docker or Docker-Compose and thus is unable to monitor anything other than containers.

$$c_{effective} = n_{used\ CPU} \cdot p_{CPU} + n_{used\ memory} \cdot p_{memory} + n_{network} \cdot p_{network} \tag{1}$$

$$c_{waste} = n_{unused\ CPU} \cdot p_{CPU} + n_{unused\ memory} \cdot p_{memory} \tag{2}$$

2.2.2 Estimating Cost

While research about estimating costs for running a cloud application does not map waste in resource provisioning directly, it does map the resources being used and the costs thereof. This makes it a very closely related topic to monitoring and calculating resource waste in cloud applications.

Development and deployment have been simplified by the introduction of serverless technologies. These serverless technologies offer a fine-granular pay-per-use pricing model. Kuhlenkamp and Klems present Costradamus [14], a cost-tracing system that enables per-request cost-tracing for cloud-based software services. It implements a generic cost model that uses three different tracing approaches: log import, response recording, and model-based tracing. Costradamus makes it possible to derive cost and performance information per API request, enabling a developer to lower waste and increase profit margins. They introduce the *Marginal Request Cost* to quantify the per-request cost. The difference between the used and metered duration of a request is taken into account in this, as well as the difference between the used and metered amount of resources. Further details about their design, implementation, and evaluation are provided but are not necessarily relevant to this research project.

The waste that Costradamus takes into account is largely due to the billing granularity of serverless functions. In December 2020 the billing granularity of AWS Lambda was reduced from 100ms to 1ms [10]. Azure Function also uses the billing granularity of 1ms [11]. This makes the waste calculations that Costradamus presented no longer useful. The concept of comparing the resources that are billed to the resources that are used does remain the same for waste calculations.

Another interesting approach to estimating the costs of running a cloud application is CostHat [15]. CostHat models the deployment costs, including compute and IO costs, of microservice-based applications. It supports microservices using *Infrastructure as a Service* (IaaS) and *Platform as a Service* (PaaS) clouds, but also serverless functions, like AWS Lambda. The model is implemented as a tool that warns developers in the Integrated Development Environment (IDE) about potentially costly code changes. This allows developers to estimate the total costs of deployment prior to actually deploying the cloud application. The total costs of deployment are calculated by taking the sum of all services. The model takes into account four cost factors: compute costs, costs of API calls, costs of IO operations, and costs of additional options (e.g. costs of elastic IP addresses). While the model is able to handle various types of microservices, it does not take the waste of resources into account.

2.2.3 Monitoring Architecture

Monitoring tools can be built using different architectures and there are various other design choices involved, such as the communication model and which type of cloud it can handle. The papers [1] [2] [3] [16] [17] compare different monitoring tools based on this.

A cloud monitoring system can have a centralized or decentralized architecture [3] [16] [17]. A centralized architecture is a single physical machine or a designated server running on a cloud node. Using a centralized architecture makes the system more vulnerable to faults. It also has an increased network load since all communication has to be done with a single node. Enough storage and compute resources need to be available at this single node. Examples of monitoring systems from the literature that use a centralized architecture are PCMONS [18], GMonE [19], and Ngmon [20], all of which are from older papers. A decentralized architecture uses multiple nodes to collect and share the monitoring data. It does not have a single point of failure and is thus much more fault-tolerant than a centralized architecture. Because there are multiple nodes each node requires less compute resources than a centralized architecture. Some examples of monitoring systems in the literature that use a decentralized architecture are NFM [21], MonSLAR [22], and MonPaaS [23], which are from more recent papers.

Monitoring tools can use different communication models, the push model, the pull model, or a hybrid model [3]. In the push model the resources that need to be monitored send the needed information to the monitoring tool without prior request. The push model can be divided into two categories: a periodic push or an event-driven push. A periodic push means the metric data is sent according to a pre-defined schedule. An event-driven push sends the metric data whenever there is a change in the data. Because the monitoring tool does not need to request the data from the resources this model eliminates the risk of a Denial of Service (DoS) attack. Examples of monitoring systems that use the push communication model are NFM [21], CloudWatch [4], and Azure Monitor [5]. The pull model on the other hand does require the monitoring tool to send requests to the various resources. This model can also be divided into two categories: a period pull and an event-driven pull. The period pull works the same way as for the push model, but the event-driven pull is used to request updated metrics based on certain events and not whenever new metric information is available at the resource. A hybrid model combines the periodic and event-driven options of the push and pull model in some form. An example of this is presented by Lin et al. [24] which uses a hybrid push protocol, combining the advantages of the periodic push and the event-driven push. Another example is Ngmon [20] that uses both the push and pull model, but for different aspects of the system.

Which types of resources or infrastructure can be monitored with a monitoring tool is not always the same. The most common type is Virtual Machines (VMs), which are monitored by launching a probe or agent on the VM. This probe sends the metric and/or log data from the specific instance to the monitoring system to be collected and processed. Most monitoring tools are capable of monitoring VMs since this is the most basic resource. Some examples of monitoring tools that can monitor VMs are CloudWatch [4], NFM [21], and MonPaaS [23]. Another type of resource to be monitored are containers. Because most monitoring tools use probes to collect data about a resource, this probe needs to be able to deal with containers, which is not always the case. More recent monitoring tools like CostHat [15] and commercial tools like CloudWatch [4] are able to handle this. NFM [21] is also able to monitor containers but does not use probes. Serverless functions are often not taken into consideration in monitoring systems [1] [3]. But there are monitoring tools developed especially for serverless functions, like Costradamus [14]. While the focus of monitoring is on compute instances, things like databases and networks can also be monitored, by Sensu [25] for example.

2.2.4 State-of-the-Art Monitoring Features

In this section, some state-of-the-art features of monitoring tools from literature are discussed.

While most monitoring tools use probes to obtain data from various resources, Suneja et al. [21] present a monitoring system without probes. This new paradigm of cloud monitoring is called *Near Field Monitoring* (NFM). Because instances are created and destroyed at a very rapid pace in cloud computing the authors of this paper came up with a way of monitoring these instances without having to launch a probe with every newly created instance. Instead, they use kernel data to collect metrics, which they have successfully tested for more than 1000 flavors of Linux.

An important aspect of using cloud applications is the Service Level Agreement (SLA) that the CSP and the user agree upon. However, the agreements in the SLA are not monitored by most monitoring tools. Shaymaa Al-Shammari and Adil Al-Yasiri [22] propose a monitoring architecture called MonSLAR that does monitor this. It monitors if the CSP and the user are holding up their side of the SLA. The system then provides the user with information about the SLA conditions being met or not. Anithakumari and Chandrasekaran [26] introduce an SLA management system that can detect and predict any expected violations. Whenever a violation is detected or predicted the system automatically adjusts the resources to reduce the number of SLA violations.

Mahmoud Al-Ayyoub et al. [27] present a multi-agent system, named *Dynamic Resources Provisioning and Monitoring* (DRPM), to manage the cloud provider's resources while taking the conditions of the SLA into account. Different agents are responsible for different tasks, including the monitoring of customers and available resources based on the customer's request. They also introduce TaskFlow to improve elastic resource provisioning for effective resource utilization. The system improves resource utilization and decreases power consumption while avoiding SLA violations.

2.3 Related Technologies

Relevant technologies are current monitoring tools and dashboard tools. The first section will discuss a selection of current monitoring tools. The second section will discuss two dashboarding tools and their technology stack.

2.3.1 Current Monitoring Tools

The cloud computing landscape is constantly evolving and changing, by offering new services or by changing how services work. This means that cloud monitoring tools need to evolve and change as well. A lot of different cloud monitoring tools are currently available, each of them having advantages and disadvantages. In this section, a selection of popular monitoring tools will be discussed.

AWS CloudWatch & Cost Explorer

AWS CloudWatch [4] is the monitoring solution provided by AWS and provides monitoring for AWS resources, applications, and services running on AWS and on-premises. The tool is used by default when an AWS account is created and collects monitoring and operational data in the form of logs, metrics, and events. All collected data can be viewed in dashboards. AWS CloudWatch can be used to detect anomalous behavior in environments, set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to keep applications running smoothly. AWS CloudWatch is a commercial monitoring tool but does offer a free tier for basic usage. Outside the free tier, the pay-as-you-go principle is adopted.

Over 30 services publish logs to AWS CloudWatch and it receives metrics from over 70 services. Containers and serverless functions also allow their logs and metrics to be collected by AWS CloudWatch. Tools are provided to set up the collection of logs and metrics and custom metrics can be created. This means that if

a cloud application is running on AWS, AWS CloudWatch is easy to set up. On-premise resources and even applications that are hosted by other CSPs can be monitored by AWS CloudWatch. The monitoring tool uses a centralized architecture for each region and uses the push model for communication.

To monitor the cost of all the resources and services used AWS offers a dedicated monitoring tool, namely AWS Cost Explorer [28]. This tool visualizes the cost and usage over time for specific services. AWS also offers separate tools for creating budgets, getting costs reports, and creating savings plans.

Azure Monitor & Microsoft Cost Management

Microsoft Azure also offers its own monitoring tool, namely Azure Monitor [5]. Azure Monitor collects, analyzes, and acts on telemetry for both cloud and on-premise environments. Azure Monitor offers tools to detect and diagnose issues for applications, VMs, and containers. It also collects logs and metrics and presents the results in dashboards. While Azure Monitor is a commercial monitoring tool free tiers are available, outside the free tier the pay-as-you-go or commitment tiers are available.

While Azure Monitor works slightly differently from AWS CloudWatch the capabilities are very similar and a lot of services from Azure can be monitored with it. Data can even be collected from cloud applications that are not hosted on Azure and be imported into Azure Monitor. If the cloud application is already running on Azure, Azure Monitor is easy to set up. Custom metrics can also be created using an API. The monitoring tool is a *Software as a Service* (SaaS) solution, so it is unknown if the architecture is centralized or decentralized. The communication model used by Azure Monitor is the push model.

Azure also offers a separate monitoring tool for cost, Microsoft Cost Management [29]. This tool shows resource usage and cost over time and lets the user create budgets and allocate costs. The tool monitors the services and resources on Azure, but it can also be used for cloud applications running on AWS. Using it for Azure is completely free, but using it for AWS is not.

GCP Cloud Monitoring & Cost Management

As the third major cloud service provider, GCP also offers its own monitoring tool: GCP Cloud Monitoring [6]. The tool makes it possible to gain visibility into the performance, availability, and health of applications and infrastructure. Metrics from resources and services hosted by GCP are automatically collected and do not incur any charges. Metrics can also be collected from AWS or other platforms, but not for free. An API for creating custom metrics is available. GCP Cloud Monitoring also offers *Ops Agent* [30], which is an agent (which is the same as a probe) that collects both metrics and logs from compute instances. The monitoring tool is a SaaS application and the communication model and architecture are unknown.

Like AWS and Azure, GCP offers a separate cost monitoring tool as well: GCP Cost Management [31]. The tool can create reports, dashboards, budgets, and alerts for the current cost trends and forecasts. It also makes it possible to optimize cloud costs and usage with intelligent recommendations. The use of GCP Cost Management is completely free, but can only be used for resources and services from GCP.

Datadog

A popular monitoring tool that is independent of a CSP is Datadog [32]. Datadog is a SaaS monitoring solution that can monitor cloud infrastructure, applications, containers, networks, logs, or serverless functions. Because Datadog requires the user to install a configuration-based agent, setting it up can be time-consuming. Datadog does not offer a free tier but is completely cloud-agnostic.

New Relic

New Relic [33] is another monitoring tool that is independent of a CSP and cloud-agnostic. It can monitor applications, infrastructure, networks, logs, containers, and serverless. Like Datadog it requires an agent to be installed, but they offer a wide range of programming languages for the agents. New Relic also does not offer a free tier but does provide rich dashboarding support.

Others

Of course, there are many other monitoring tools available, like Sumo Logic [34], Splunk [35], Nagios [36], and Sensu [25], just to name a few. While these tools have their advantages and disadvantages they are not different enough from the tools already discussed to add anything to this section.

2.3.2 Dashboards

Cloud monitoring tools use dashboards to present the data to the user. A dashboard consists of a set of visualizations of data, which can be in the form of a graph, a heatmap, a pie chart, or a table. A cloud monitoring dashboard usually has multiple pages giving information about different specific aspects of the cloud application. While the monitoring tools discussed in Section 2.3.1 all combine the collection of data with a dashboard, this can also be separated. In this section, two popular stacks for creating dashboards for cloud monitoring will be discussed.

The first stack for creating a monitoring tool with a dashboard is Grafana [37] and Prometheus [38]. Grafana is a well-established dashboarding tool and allows users to quickly create dashboards to their specific requirements. It allows the user to query, visualize, and alert on metrics, no matter where they are stored. Grafana integrates with various other technologies, among which is Prometheus [39], allowing for an easy setup. Prometheus is an open-source monitoring system developed by engineers at SoundCloud [1] to move toward microservice architectures. Prometheus is a time-series database that collects and stores metric data using a pull model. Because Grafana and Prometheus are easily configured together they make a very useful stack for creating custom monitoring systems with a dashboard.

An alternative stack is Kibana [40] and ElasticSearch [41]. Elasticsearch is a distributed, RESTful search and analytics engine, which centrally stores the data. Kibana allows the user to visualize Elasticsearch data in the form of a dashboard. Because Kibana is specifically created for ElasticSearch they are very easy to use together and offer a good alternative to the Grafana and Prometheus stack.

2.4 Conclusion

In this chapter, the necessary background information about the reasons for cloud monitoring and cloud monitoring properties are provided. The related literature is discussed and while there is not a lot of literature about the waste of resources specifically, there is literature on closely related topics. The earlier works that this research project is building upon offer stepping stones for creating a formal definition of waste and the papers about estimating cost help in understanding how the cost of running cloud applications can be estimated. The different monitoring architectures and communication models are explained. In the last section, a selection of current monitoring tools and dashboarding tools are discussed. Together this gives the needed information to move on to preparing formal interviews in the next chapter.

[1] SoundCloud is a platform that allows users to share their own music, `https://soundcloud.com/`

3 Interviews

In order to get a deeper insight into how developers use cloud monitoring tools in practice, formal interviews with developers from various projects and companies have been conducted. The purpose of these interviews is to find out what aspects of cloud monitoring tools work well and are used frequently, as well as finding out what the limitations are of the used monitoring tools. A secondary goal is to find out if developers think a cloud waste monitoring tool has added value and how such a tool would work best. The research question for the purpose of these interviews is: *"How is cloud monitoring handled for different cloud applications?"*

3.1 Methodology

To conduct the formal interviews systematically the 5 steps that are outlined by Runeson and Höst [42] will be used. These guidelines are specifically designed for case studies in the software engineering field. This makes them easier to work within the scope of this project than more general, and thus broader, guidelines for case studies [43]. The steps are as follows:

1. Case study design
2. Preparation for data collection
3. Collecting evidence
4. Analysis of collected data
5. Reporting

The first two steps will be discussed in the following section. The *Collecting evidence* step is the actual execution of the interviews. The *Analysis of collected data* and *Reporting* steps will be discussed in Section 3.3 and 3.4.

3.2 Case Study Design & Preparation

3.2.1 Objective

The objective of conducting formal interviews with experts in the field of cloud engineering is to get an understanding of how different cloud applications and their monitoring tools are handled. This is needed to define the concept of waste in the context of resource provisioning in cloud applications, but also to gather information about how developers handle monitoring tools, and to create requirements for a waste monitoring system. During the interviews, the architecture of the projects, used monitoring tools, the process of resource selection, cost management, and wasted resources are to be discussed. This makes it possible to answer the research question for the interviews: *"How is cloud monitoring handled for different cloud applications?"*

3.2.2 Methods

The interviews are exploratory in nature and are thus conducted in the form of a *semi-structured interview*, which is defined as:

> *"The interviewer has an interview guide that serves as a checklist of topics to be covered and a default wording and order for the questions, but the wording and order are often substantially modified based on the flow of the interview, and additional unplanned questions are asked to follow up on what the interviewee says."* [43]

This means a set of questions for the interviews were prepared for this purpose, which can be seen in Appendix A. While all of these questions were asked in some form, it was important to ask follow-up questions relating to the answers given by the interviewees. Because of this, the transcribed interviews have slightly different questions every time.

The interviews were recorded and transcribed to process the results. Since the interviews were conducted while there were still many COVID-19 measures in place it was deemed best to conduct them digitally. This also made it easy to record the interview with both audio and video. The recorded interviews were transcribed and sent back to the interviewee for verification. This was to ensure that everything they said and the way it is transcribed is in accordance with their opinion. The transcribed interviews are available on demand.

3.2.3 Demographic information interviewees

Because this thesis project is being done in collaboration with the *Cloud Engineering* department of Deloitte, it was possible to get in contact with a number of experts in the field of cloud engineering. The selected experts are working at Deloitte for a client or working at a client company directly. A total of 6 experts were selected to be interviewed, based on their expertise and current projects. All of the experts are working on different projects for different clients, creating a broader insight into how different projects are handled. Their current project will be the focus of the interview, but their experience from past projects will be taken into consideration as well. The projects involved are varying in scale, architecture, and maturity, thus giving a broad insight into how developers handle different aspects of cloud engineering.

Half of the experts have a **master's degree**, whereas the others have a **bachelor's or an applied sciences degree**. Half of them did their studies in a computing-science-related field, while the others switched fields during their careers. All experts are in **senior or management/technical lead positions**. They have at least **5 years of experience in the field**, with **a total of 54 years of experience between them**. During their time at their current employer, they have done **at least 3 projects**, some have even done **over 20 projects**, giving **a total of over 46 projects**. Depending on the expert, they have more expertise in working with **certain CSPs, architectures, and type of applications**, which further diversifies the knowledge they possess as a group and reduces bias.

For privacy reasons, the names of the experts have been anonymized and the companies for which they are doing a project have been omitted.

3.3 Analysis

To analyze the transcribed interviews the Constant Comparison Method as described in [44] was used. This means quotes had to be extracted from the interviews and codes needed to be assigned. All quotes can be found in Appendix B and the codes are introduced in Section 3.3.2.

3.3.1 Codes hierarchy

The codes with their hierarchy can be found in Figure 2. Each quote has at least one code assigned, but can also have multiple codes. If a code is attached to another code above it in the hierarchy, the code above is implicitly assigned as well. For example, if the code *"3rd Party Monitoring Tool"* is assigned to a quote, the codes *"Monitoring Tool"* and *"Monitoring"* are implicitly assigned. The codes *"Datadog"* or *"Splunk"* are below it and are thus not implicitly assigned. After each code, the number of occurrences is stated between square brackets. If a code has one or more codes attached to it below it, there will be two numbers. The first is an aggregated number of occurrences that takes into account all the time the code was implicitly assigned

and the second is the number of occurrences where the code itself was assigned. The codes that have a blue box are directly related to requirements (see Section 4.1).

3.3.2 Code Occurrences

In Figure 3 each code that was assigned to at least one quote is shown, ranking from most occurrences to least. This is the non-aggregated count of occurrences, meaning that a code that has other codes under it in the hierarchy (see Section 3.3.1) is only counted if the code itself is assigned. Figure 4 shows the aggregated count of occurrences, thus also counting the implicitly assigned codes. For instance, *"Monitoring"* has a lot of codes underneath it in the hierarchy and for each time one of those codes was attached to a quote the code *"Monitoring"* is also counted. This way the more general code is implied by the more specific code. The difference between Figure 4 and Figure 3 is quite large due to this. The more general code is often not assigned to a quote directly because a more specific code can be used.

From the aggregated code occurrences it is clear that most of the quotes are related to *"Monitoring"*, which is due to the main research question of the interviews: "How is cloud monitoring handled for different cloud applications?". Going down the list of codes shows which codes came up most often, which gives a good indication of how important each code is. The results of the interviews are discussed in Section 3.4.

3.4 Findings from interviews

In this section the findings from the interviews based on the quotes (see Appendix B and the codes assigned to them will be presented. It will follow the structure of the codes as presented in Section 3.3.1

- **Monitoring**
  - **Monitoring Reason**
    The reasons to monitor in general always comes down to making sure that the application is running as it should. This includes checking for errors, bugs, long-running requests, and performance issues in general A11 B19 C24 D14 D18 F22.
  - **Monitoring Tool** (incl. "Reason for Specific Tool", "Provider Native Monitoring Tool", "3rd Party Monitoring Tool", "Datadog", and "Splunk")
    During the interviews, it has become very clear that most projects use the monitoring tools provided by the cloud service providers A13 B6 C37 E23 F29. In the (recent) past this was not the case and due to the limitations of the monitoring tools offered by cloud service providers a third-party tool was often used C37 E20 E21 F39. The most common reasons for not using the CSPs' native monitoring tools were due to the limitations of the native tools and the time it took and the difficulty of setting it up C26 F28 F40. Datadog was by far the most popular third-party monitoring tool amongst the interviewees, with 4 out of 6 interviewees having worked with it in the past and one that still works with it C19 D16 E21 F28. The one project that still uses Datadog instead of the CSPs' native monitoring tools does that due to project constraints D13 D16 D20 D24. Splunk is a tool that did come up in two projects, which offers additional features for security monitoring and error log handling D16 D17 E22. Various other monitoring tools have been used in the past where each tool had a certain advantage over another. Because of the improvements made to the CSPs' native monitoring tools over the recent past, the third-party tools have been slowly phased out for most projects D16 C28 C37 E23 E24 F29.
  - **Cost Monitoring**
    Cost monitoring is always done to a certain degree, but as long as the total cost is relatively stable over the months there is no real incentive to optimize D14 B13 C20 C32. Some developers do take a look at the bills and see if they can optimize something, but that depends on the individual and available

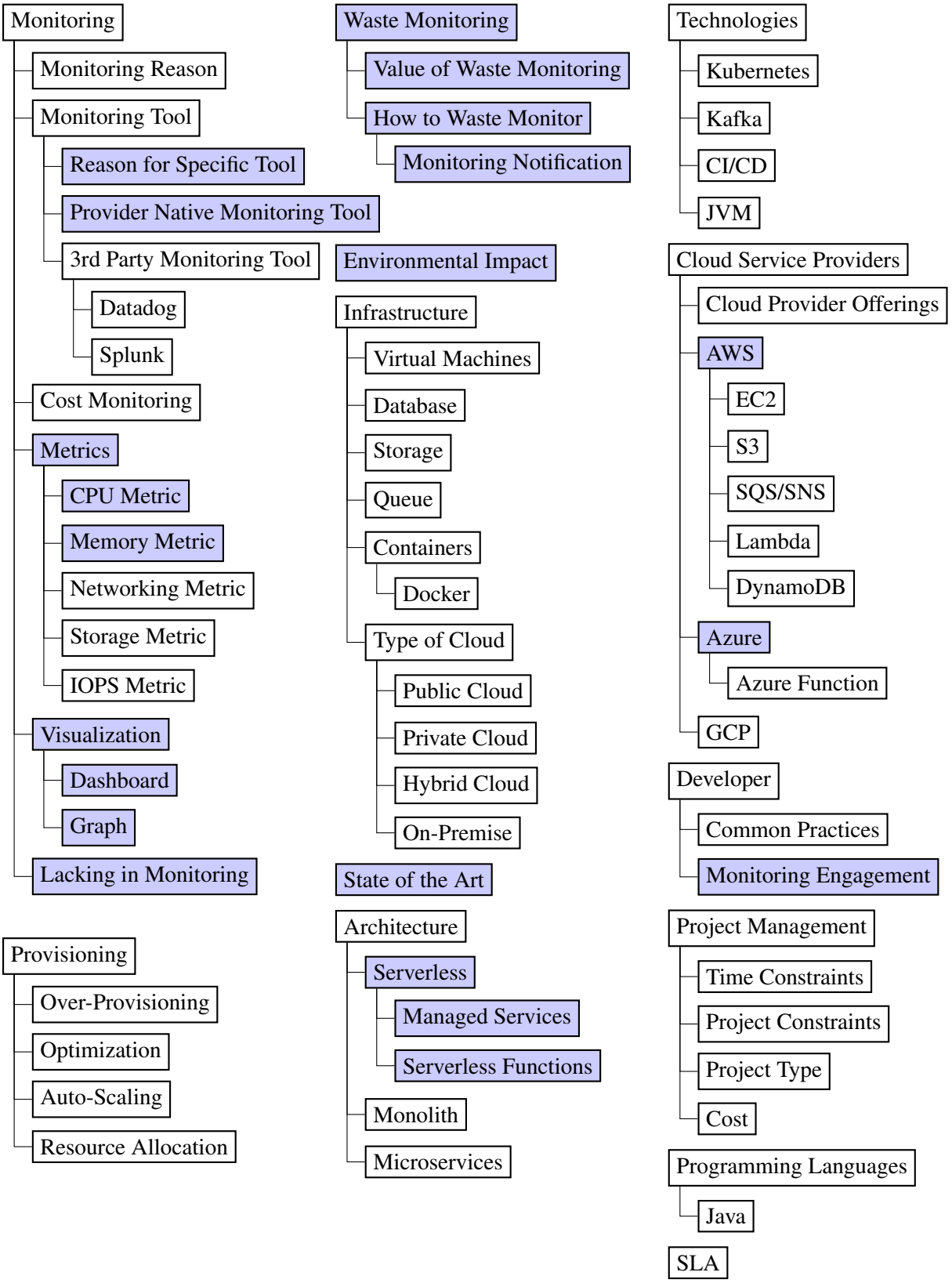


Figure 2: The hierarchy of all the codes used in the analysis of the interviews.

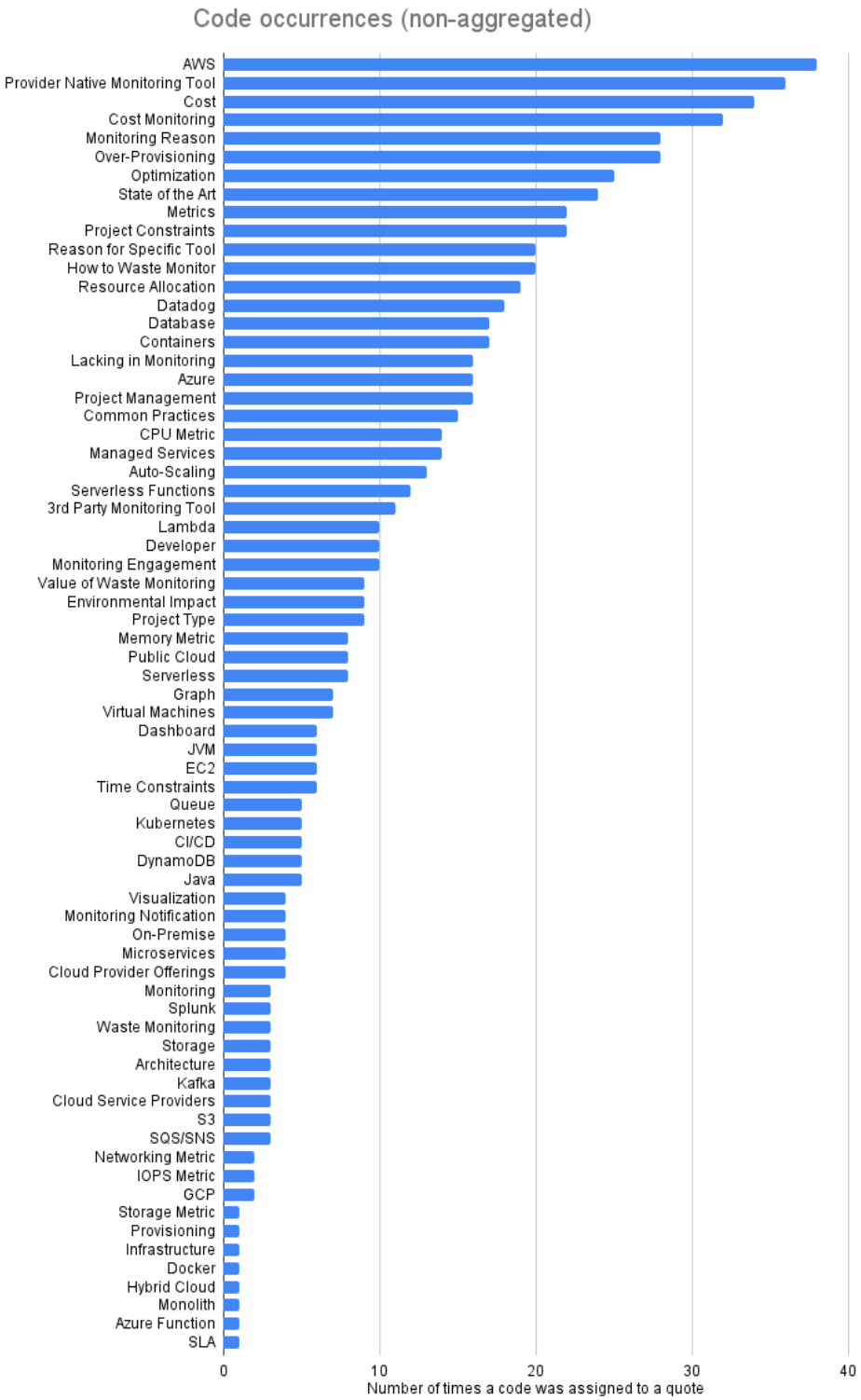


Figure 3: Number of times a code was explicitly assigned to a quote, so the results are non-aggregated.

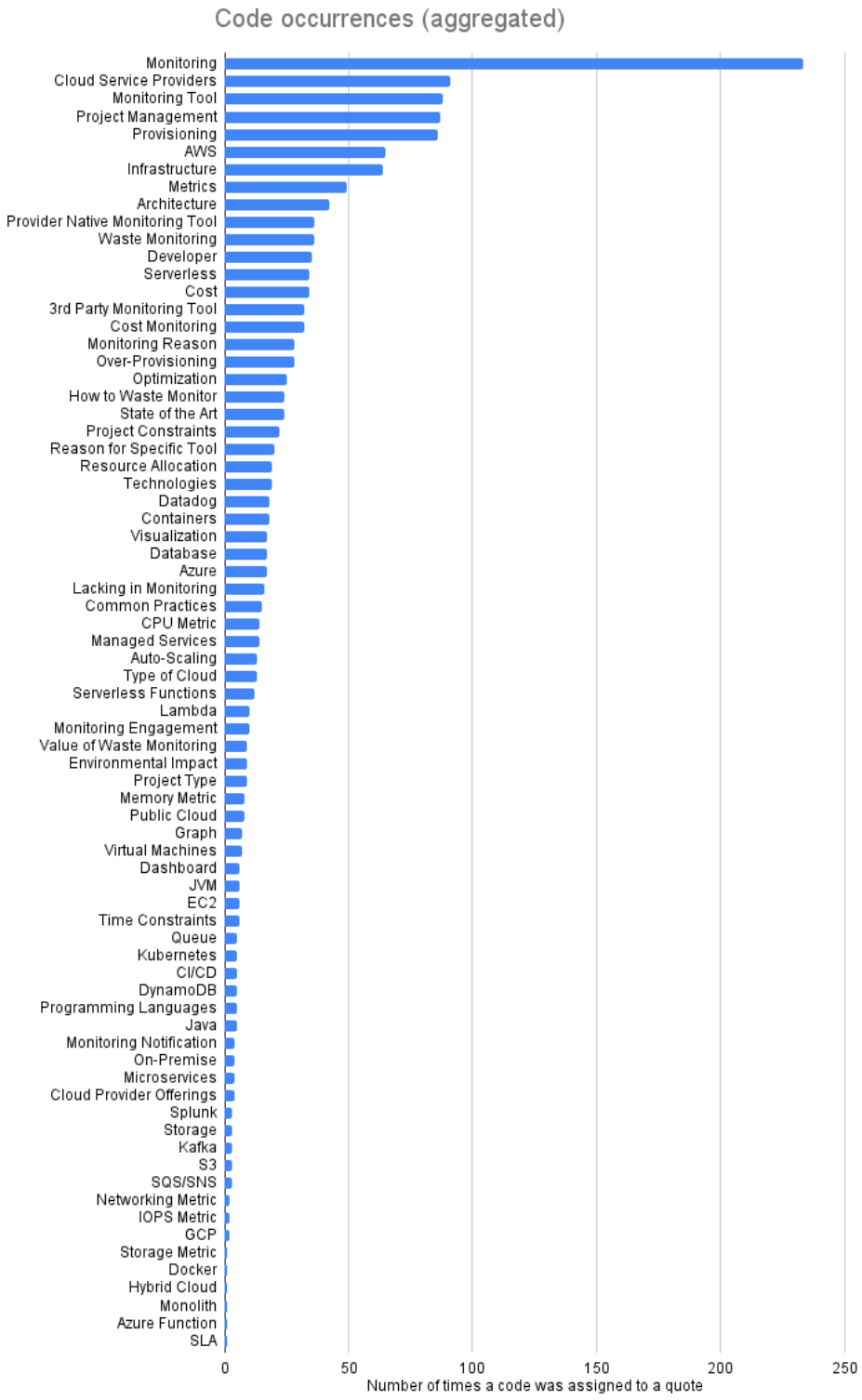


Figure 4: Number of times a code was assigned to a quote, both explicitly and implicitly, so the results are aggregated.

time B19. However, if the costs are suddenly going up, developers are asked to find out why this is happening and to optimize it A15. So unless the costs are unexpectedly high, the cost monitoring dashboards generally do not get a lot of attention A14 B20.

Precisely pinpointing what is causing an increased cost is something that can be difficult and stands to gain from some improvements B18 C32

The cost monitoring tools used are almost always the native CSP tools, even when the project is using a third-party monitoring tool for other monitoring purposes C20 F36 E25 D21.

- **Metrics** (incl. "CPU Metric", "Memory Metric", "Networking Metric", "Storage Metric", and "IOPS Metric")
  The metrics that are considered most important are all related to the performance of the resources. The most important metrics are the CPU and memory metrics B9 C19 C24 E26 F33. Other metrics such as metrics about networking, IOPS, and storage are also of importance but are mentioned considerably less often than CPU and memory F33 F35 C44 E26. The "CPU Metric" and "Memory Metric" codes are assigned a total of 22 times, whereas the codes "Networking Metric", "Storage Metric", and "IOPS Metric" are assigned a total of 5 times. For cost monitoring, the metrics about CPU, memory, and storage are the most important E26.

- **Visualization** (incl. "Dashboard" and "Graph")
  The preferred method of visualizing metrics is as a graph C35 D24 E28 F55. Presenting such graphs in a dashboard is also preferred by most B26 E34 F69. As stated under "Monitoring Tool", a waste monitoring tool should integrate with the already used (ISP's native) monitoring tool. Seeing as most of these tools already use dashboards with graphs this is a very logical choice for waste monitoring as well.

- **Lacking in Monitoring**
  The most mentioned feature that is lacking in monitoring tools currently being used is an aggregate overview of all the nodes that are running A19 A20 B22 B24 E30. Presenting a general picture of the whole architecture and allowing a user to zoom into more specific sections. For instance, zooming into a database and then being able to zoom into a specific deployment instance or node of that database. Showing the logs and metrics (where available) for each level of zoom. Others are quite satisfied with the current capabilities C36 D25. One interviewee mentioned that they used to have a long list of things that were missing in what AWS provided, but that there has been a lot of improvement in recent years F57.

  While it is good to know what developers would like to see added to the capabilities of monitoring tools, this is outside the scope of this project.

- **Waste Monitoring**

  - **Value of Waste Monitoring**
    A waste monitoring tool was generally seen as a useful addition to the currently available monitoring features A21 C39 D26 E33 F61 F64. Especially emphasizing the potential cost savings and positive impact on the environment are important features C39 D26 E33 F61 F64. Having a waste monitoring tool available is also a selling point to (potential) clients.

  - **How to Waste Monitor**
    When asked in what way a waste monitoring tool would be most useful, most interviewees responded that a continuously running tool available in a dashboard or graph would be most useful A25 B26 D27 E34 F69. Integrating a waste monitoring system with the currently used monitoring tools would be the best option

for this B25. Whereas integrating a waste monitoring tool in the CI/CD pipeline was not considered useful or at least not a priority, but could work F67 B27 E35. A single-run script to see if an application has waste was not considered useful because the amount of waste over time is essential and only getting this information for a certain point in time is not enough information A23.

  * **Monitoring Notification**
    The interaction with monitoring tools is rather limited and it usually requires something to go wrong or incur raised costs for a developer to actively look at it. This means that simply making the information available in a dashboard would likely go unnoticed for a significant amount of time. While this is fine if there is little waste, it would definitely produce better results if developers are notified whenever there is significant waste or after a certain time period A21 A25 C42.

- **Provisioning**

  - **Resource Allocation and Over-Provisioning**
    When asked what the process was for the initial resource selection for a project, a lot of interviewees said that they simply take something that they know will be able to handle the project F12 C10 E8 E12 E14. With a preference for specific resources they have worked with before. While this can obviously cause over-provisioning, it is hard to determine the right initial resources. When possible, expected traffic or estimated loads are used to determine which resources to select A7 C12 E12. While this is difficult to do for greenfield projects, projects being rebuilt for the cloud have traffic and load data available. The projects where an existing solution is simply being ported to the cloud are usually very bad for selecting the right amount of resources since they usually have been over-provisioned before moving to the cloud F66.

  - **Optimization**
    Optimization is usually not a big concern as long as the application does what it is intended to do in a reasonable amount of time. Optimizations are done later if time allows E10. The optimizations that do happen are usually triggered by higher costs than normal. A common optimization technique is to combine resources (to an extent) for the development, test, and acceptance environments B5. Other kinds of optimizations are focused on reducing the complexity in the environment C14, replacing the memory heavy JVM (Java Virtual Machine) with a different solution to run Java code C16, or even switching from an x86 architecture processor to an ARM processor, which requires a lot less wattage to run and thus would reduce the environmental impact of the application C43.

  - **Auto-Scaling**
    Auto-Scaling is used mostly for CPUs since that is the bottleneck in most systems B7. It does have the issue of not being able to scale fast enough when loads suddenly increase E27. For one project that had heavy peak loads, this was not fast enough and thus required timed-based scaling rules. This resulted in having intentional over-provisioning to handle these spikes. B15 B16.

- **Infrastructure**

  - **Virtual Machines**
    During the interviews, it turned out that Virtual Machines (VMs) are used considerably less than in the past. When using managed services or running containers the compute hardware still has to be provisioned manually, which comes down to provisioning virtual machines C3 F3. Virtual machines are still being used a lot C17, but they are no longer the first choice for most projects D9.

  - **Database**
    A wide variety of databases is used across the different projects, including Neptune, Redshift, RDS, DynamoDB, DocumentDB, and Cosmos DB A1 C5 D12 F5. Managed databases are used quite a bit,

but the size of the database still has to be set by hand, thus leaving room for waste C11. Which database (type) is used is not of great importance for the scope of this thesis.

- **Storage**
  While storage is needed in most projects it was not mentioned a lot during the interviews. The interviewees that did mention it, stored their objects in AWS S3 C5 C7 D4. The amount of storage chosen is always based on the expected growth of the data that needs to be put into it, so it is highly unlikely to have the exact amount provisioned that is needed. Because of this, it is highly debatable what exactly would be the waste in storage.

- **Queue**
  Due to projects opting for serverless cloud solutions in recent years, a loosely coupled microservice architecture is needed (see "Microservices)". This means that the components can act independently from each other, but still have the need to communicate for certain things. This also allows for a component to be switched without affecting the performance of other components in the system. To facilitate the communication between the different components of a system a queue can be used E4 F6. This allows for messages to be handled asynchronously.

- **Containers** (incl. "Docker")
  While serverless is becoming more standard, containers are still a very popular way to run applications in the cloud B28 C2 D3 E3 F2. This is done in different ways, but the most popular ones are the container services provided by the CSPs or even the managed container services B28 C2 D3. Containers tend to have a certain amount of waste by default because it requires an environment around the actual application for each container to be spun up E7 E11.

  Containers are often used in combination with Kubernetes (see "Kubernetes").

- **Type of Cloud** (incl. "Public Cloud", "Private Cloud", "Hybrid Cloud", and "On-Premise")
  The projects almost always use public cloud A3 B3 C8 D2, with a few small exceptions E6 F7. Some projects require compliance with certain regulations, which in turn requires some information to be hosted on-premise. This means that the cloud applications themselves run completely in the public cloud but do exchange data with these on-premise servers C8. Another exception is for projects that are moving their on-premise applications into the public cloud, during the transition phase there can be the need to utilize both the old on-premise infrastructure and the new public cloud resources, resulting in a temporary hybrid cloud E6. This is always a transitional phase and the goal of these projects is to have everything in the public cloud.

- **State of the Art**
  The state of the art in cloud computing has gone through quite a lot of phases already. It used to be a lot of Virtual Machines, over time that changed to a preference for using containers, and now developers use managed services and serverless functions when possible, creating a serverless mindset. Of course, all of these architectures are still used, but the focus has shifted to managed services and serverless functions in recent years B28 C4 E2 E15 E16 E37 D3 D9.

  The way the cloud is used and can be used is constantly changing. This forces developers to continuously be informed about the current state of the services they are using D31.

- **Architecture**

  - **Serverless**

    * **Managed Services**
      As mentioned in the previous sections, a lot more managed services have become available in recent

years. This means letting the CSP do more of the work automatically, which saves on man-hours, making it very popular C2 C4 D3 E4 F4. Managed services are used for all kinds of things, like containers, queues, databases, Kubernetes, etc. Some managed services take care of the resource provisioning, including the scaling, so any potential waste that occurs here is not something that can be reduced E9.

  * **Serverless Functions**
    Serverless functions are also being widely utilized A1 C33 D3 E4 F6. While serverless functions need to be customized to the CSP, both the projects on AWS and Azure are using them. The fact that serverless functions have very little waste and thus reduce cost in most cases is a big factor in deciding to use them C33 F72 F73.

- **Monolith**
  A monolithic application is self-contained and independent from other applications. This means it does not have separate components for frontends, backends, or microservices. As such it is very different from a serverless application and only one interviewee was working on a project that used a monolithic application B1.

- **Microservices**
  Microservices are loosely coupled components that, together, form a complete application. This allows for components to be switched or changed without affecting the other components. While it is possible to run this type of system using regular virtual machines, it also allows for components to be managed services. A serverless stack is composed of different microservices. This means that microservices are a very popular architecture to use E1 E3 F3 D3.

- **Technologies**

  - **Kubernetes**
    Containers are quite a popular technology to use amongst the interviewees, as described under the "Containers" section. In most cases, the containers are run using Kubernetes to manage all the individual containers C2 F2. AWS offers a managed Kubernetes service, as well as Fargate, which does not require EC2 instances and handles the provisioning aspect as well [45]. Fargate is also being used in one of the projects D3.

  - **Kafka**
    Apache Kafka is a distributed event store and stream-processing platform. This is an alternative to the regular queue (see "Queue"). Kafka allows for massive streams of data in a fault-tolerant way. Kafka is also offered as a managed service by AWS (MSK), which is used by two of the projects C6 F4.

  - **CI/CD**
    CI/CD (Continuous Integration and Continue Deployment) is being used in at least one of the projects F7 and likely more. Integrating a waste monitoring tool into the CI/CD pipeline was not seen as useful, but has the potential to work B27 E35 F67.

  - **JVM**
    A JVM is used to run an application written in Java. The JVM is notorious for being very memory-hungry C10 E13 F15. This influences the choice in resource allocation for an application C10 F15. While there are alternatives to using a JVM to run a Java application, these are currently not being used by the interviewees' C16.

- **Cloud Service Providers**

- **Cloud Provider Offerings**
  All cloud service providers are constantly improving and expanding the services they offer. Especially the "Managed Services" (see the corresponding section) have seen many new additions and improvements in recent years E16. The level of service a CSP provides can sometimes be a reason for intentionally over-provisioning a resource A4 B4.

- **AWS** (incl. "EC2", "S3", "SQS/SNS", "Lambda", and "DynamoDB")
  AWS is the most popular CSP amongst the interviewees, with four out of six interviewees doing their current project on it C1 D1 E5 F7. More serverless options are becoming available in recent years, mostly in the form of managed services. The possibilities of the different services are also expanding and the user-friendliness is increasing C37 E23 F29 F32.

  The individual services are covered in the sections about "Infrastructure" and "Architecture". The use of these services does show that quite a variety of AWS services are being used in projects and monitoring them requires collecting the data in AWS CloudWatch. In the aggregated code occurrence graph, presented in Figure 4, AWS is sixth, while in the non-aggregated code occurrence graph, presented in Figure 3, AWS is ranked first. This shows that the interviewees are using AWS and its various services a lot.

- **Azure** (incl. "Azure Function")
  Azure is less popular than AWS amongst the interviewees, with only two out of six using it for their current project A13 B2 B3. Azure is also continuously adding more services and additional features. One of the interviewees is also using the Azure Function for their project A1.

  Azure ranks considerably lower than AWS in both the non-aggregated as the aggregated code occurrences graphs (see Figures 3 and 4), which is partly due to only two out of the six interviewees using Azure for their current projects. But it striking that the services Azure provides are not mentioned by name anywhere near as much as is the case for AWS.

- **GCP**
  GCP is not actively being used amongst the interviewees and is seen as an upcoming provider E5. Of course, this does not necessarily reflect the cloud engineering industry as a whole, but for the scope of this project, GCP will be left out.

- **Developer**

  - **Common Practices**
    Developers have common practices when it comes to how they decide on resource provisioning, monitoring engagement, how they deal with costs, and various aspects of monitoring. While these common practices do tend to differ quite a bit for each interviewee, it is clear that aspects mentioned are handled according to what a developer is used to. See the sections "Resource Allocation", "Monitoring Engagement", and "Cost Monitoring", for more details about the various common practices.

  - **Monitoring Engagement**
    The way a developer deals with monitoring tools is largely depending on the individual E39 D32 B20. Some developers like to optimize and keep everything in check and will make the extra effort to do so. Whereas other developers do not care that much personally and are guided by the project constraints/-time constraints. If there is no focus on monitoring in detail from the project management, dashboards will go unchecked for longer periods of time A14 D32. Making sure developers engage with the new waste monitoring tool will be important. See also "Monitoring Notification" under "Waste Monitoring".

- **Project Management** (incl. "Time Constraints", "Project Constraints", "Project Type", and "Cost")

The way a project is managed and the various constraints that are imposed can have a big impact on a project and greatly influence a developer's decisions a great deal. Time constraints are one of the biggest factors because developers have to make deadlines C26 D22 D25 E10. In order to make those deadlines, they cannot always spend as much time on something as they would have liked to or can not add features they would like to add. Specific project constraints are almost always a factor in decision-making as well. Some projects need to be able to handle extreme peak loads at certain times B16 B21 F53 F54. Various other agreements can create project constraints, further influencing decision-making in the development of an implication. The type of application that is to be created is of course a massive factor in the design of the project. One of the interviewees was working on a monolith application because that made the most sense for their use case B1. Due to that, serverless functions are no longer an option. It depends on the project if cost is a big factor in decision-making. For some projects, the cost of running the cloud resources is negligible compared to the total costs C13 F9 F18, or there are factors, like performance, that are more important than cost B5. But for other projects, it is important to keep the cost low, while making sure the performance is where it should be A8 D10 D11 D12. All of these factors can greatly influence the way a developer is making decisions while designing and building an application. If waste is detected in an application this information is very important to pinpoint where the waste can be reduced and where the waste is unavoidable due to constraints.

- **Environmental Impact**
  Within Deloitte, there is a big drive to try to be as environmentally friendly as possible E33. The waste monitoring tool idea has been very well received because of the environmental angle to it. This is something that helps them sell to their clients as well, so it should translate to other companies.

  Due to the lack of incentive coming from a cost perspective (as explained in the "Project Management" section) the environmental perspective is very important to create an incentive to reduce waste. By making developers and clients aware that they are running resources that consume power, which hurts the environment, this incentive to reduce waste in resource provisioning can be created F62 F63 F64 E33.

- **Programming Languages** (incl. "Java")
  The only programming language that came up was Java, which was always in regards to the JVM that is being used to run the applications and the amount of memory needed for it. While other programming languages were never mentioned in regards to decisions about the cloud infrastructure, Java was mentioned in 3 out of the 6 interviews. This is due to the use of the "JVM" to run those applications, which did impact choices regarding the infrastructure C10 F15. See "JVM" for more.

- **SLA**
  While Service Level Agreement (SLA) is only mentioned once E18, this is something that every cloud engineer has to keep in mind during development. What the companies agreed upon in the SLA needs to be realized and checking that is something that has to happen from time to time. Monitoring tools can be used to get data about whether or not the agreements in the SLA are being realized.

3.5 Comparison between Literature & Findings

During the analysis of the interviews, some interesting differences between the findings and the literature, as summarized in Section 2.2, were encountered. The field of cloud engineering, and computing science as a whole, is constantly evolving at a rapid pace, so the literature may be somewhat behind. The commercial field and the academic field do not necessarily have the same way of looking at certain developments. In this section, the big and unexpected differences will be discussed.

Infrastructure

The first big difference is the infrastructure that is being used in the development of cloud applications. The literature about monitoring tools mostly discusses monitoring VMs and containers, but the interviewees are using serverless technologies most of the time D2 C4 E15 F48 B28. Their serverless mindset means that they will use serverless functions and managed services by default and only revert to containers and VMs if the specific use case does not work with serverless technology. While serverless technology is discussed to an extent in the literature [14] [15] this mostly concerns serverless functions. Managed services are used a great deal since they save time and thus costs. This difference might be due to the literature being a couple of years old [1] [2] [3] [16] [17] and papers discussing monitoring managed services have yet to be written. Since monitoring a managed service is not that different from monitoring a VM or container, which is being used underneath, it could also be possible the topic is not important enough to publish a paper on.

Monitoring Architecture

Another unexpected difference was regarding the architecture and design of monitoring tools. A lot of papers [3] [16] [17] discuss and compare the different architectures that are used and different design choices that have been made for various monitoring tools. However, the interviewees do not really care how the monitoring tool is realized, as long as the presented data is secure, accessible, and correct C30 C31. This is likely to be the difference between the commercial field and the academic field. If a tool is doing what it is intended to do, that is often enough in the commercial field. Whereas in the academic field there is a much larger focus on how the end result was achieved and why certain design choices are made.

Integration

The next big difference between the literature and the findings from the interviews is that in the academic field a new tool or system is often developed as a stand-alone tool, but the interviewees actually prefer a new tool or system to be integrated with the tools they are already using B25. This is mostly because having to look at multiple monitoring tools would take a lot of time for the developer. In academics, it makes sense to demonstrate something new in cloud monitoring by building a tool for it. Not integrating with something like AWS CloudWatch or Azure Monitor keeps the focus on what they are adding to the existing literature.

Optimization

Another difference is in how optimization is handled. Cost optimization is often not a high priority as long as the bill is not unexpectedly high A15 D22 F17. A common optimization that is being applied is combing resources for the development, test, and acceptance environments B5 C15. Reducing the complexity of the system is seen as a more important optimization than cost optimization by some C14. Interviewees also talked about how memory hungry the JVM is and that there are optimized alternatives possible C16 E13. Switching from an X86 architecture to an ARM architecture would reduce the power consumption, which would be another possibility for optimization C18. The last two optimization suggestions would take a lot of time to execute and are thus not very viable options C43. The literature about optimization focuses on automatically optimizing resources based on data [27] and cost optimizations [14] [15]. While these might be valuable optimization tools, developers do not want to use a lot of different monitoring systems and are thus not using these systems. If these concepts would be integrated with the monitoring tools developers are using they might start utilizing these features. So, the literature and the interviewees focus on very different aspects of cloud computing when talking about cost optimization and optimization in general.

Features

The features discussed in Section 2.2.4 never came up during the interviews. The NFM [21] monitoring that does not use probes is likely not important to the interviewees, since the architecture of a monitoring tool was not seen as important as discussed earlier in this section. The monitoring architecture MonSLAR [22] that checks, predicts, and takes action upon SLA violations also never came up. Making sure the SLAs are kept did come up once during the interviews E18, so there may be different systems in place for this. The *Dynamic Resources Provisioning and Monitoring* [27] (DRPM) system did not come up during the interviews either, which might be due to it focussing more on the CSP's perspective of monitoring. The cloud computing landscape is rapidly changing and the mindset of developers changes along with it, as discussed at the beginning of this section. This might cause certain features to become less important due to new developments.

Waste Monitoring

While there was not a lot of literature about waste monitoring, it was seen as a very useful tool by the interviewees. The concept of monitoring waste in cloud applications is not that recent, since the prior works are from 2018 [12] and 2019 [13] but was new to most interviewees. The interviewees saw great value in creating a system that could monitor waste. The main drivers are the reduced costs and the reduced environmental impact of a cloud application D26 E33 F61 F62 F64. The different interests and priorities of the academic world and the commercial world might be the reason for the different levels of interest in waste monitoring.

4 Waste Monitoring System

In this section, the various aspects of the *Waste Monitoring System* will be presented. First, the requirements for such a system are defined based on the findings from the interviews. The next section is about formally defining waste in the scope of this research project. This will consist of a *Waste Ratio* that expresses the waste as an exact figure and *Potential Savings* that expresses the waste as a monetary value of potential savings. Lastly is the *Design* section, where the architectural framework that will outline how a waste monitoring system should be built is presented. Since there is a lot of data being transferred between the different components, a data flow diagram is presented to further clarify this. To give a visualization of the information that will be presented in a waste monitoring dashboard a set of mock-ups will also be presented.

4.1 Requirements

Using the analysis and the findings of the interviews requirements for the system to be developed have been formulated. First, the functional requirements are presented, the non-functional requirements are presented afterward, and finally, some requirements that were taken into consideration but are deemed unsuitable for the waste monitoring dashboard. The functional requirements are presented in order of importance based on the findings from the interviews, except for the 6^{th} requirement. This requirement is of lower importance because the system can be built for a specific application that uses one specific architecture and thus does need to be able to handle different architectures. This is of course still important because the architecture of a system might change over time. The requirements are as follows:

1. **Integrate with currently used monitoring tools**
   Integration with the tools already being used is something that came up in almost all interviews. Developers do not want to have to run a separate program to monitor additional metrics and want to have the ease of use by having it integrated with the tools they are already using. These are the provider native tools in most cases, AWS being used the most in the projects discussed and Azure being a good second candidate. Datadog was used a lot in the past but is not being used much anymore, thus making it an unsuitable candidate for the scope of this project.

2. **Waste should be monitored based on CPU and memory metrics**
   CPU and memory are by far the most important metrics for developers and are most likely to cause unexpected costs. Performance metrics in general are named a lot, so expanding to other metrics that relate to performance would be a good addition if time allows. This includes, but is not limited to, IOPS for databases, storage metrics, and networking metrics.

3. **Results should be presented in graphs and dashboards**
   This is pretty much the consensus amongst all the interviewees. Simple graphs in a dashboard will go a long way to make the data easy to understand or even to present to stakeholders.

4. **The connection between reducing waste and cost savings should be made clear to users**
   The money saved by reducing waste is an incentive for developers to take action, but also for projects to free up time for developers to work on it. It is very useful to give estimates on possible savings, so the amount of waste can be translated to cost.

5. **Generate reports on a schedule or whenever there is significant waste**
   The interaction with monitoring tools is rather limited and usually requires something to go wrong or incur increased costs. This means that simply making the information available in a dashboard would likely go unnoticed for a significant amount of time. While this is fine if there is little waste, it would produce better results if developers are notified whenever there is significant waste. To make

sure that developers are aware of the amount of waste in their applications, reports should be generated automatically after a certain time period. This report should give an overview of the waste of resources since the last report.

6. **It needs to be able to handle applications using various architectures and types of infrastructure**
   Projects are moving towards a serverless landscape, so that should be handled. This includes managed services, but also serverless functions. The latter will probably not produce a lot of waste, so the focus should be the managed services. Projects still use containers a lot, so that needs to be handled as well. Virtual machines are a lot less popular than they used to be and are often used for managed services. This means that the way the virtual machines are handled can be very different from the regular way (where the user has complete control). It should be monitored nonetheless.

7. **[Optional] The connection between reducing waste and reducing the environmental impact should be made clear to users**
   The environmental impact of waste is a big selling point for waste monitoring. Especially for projects where the cost of the cloud is not an issue, of which there are plenty. Companies often have reducing their environmental impact high on the agenda and a waste monitoring system would help them with realizing such goals. This requirement is optional because the environmental impact reduction is an incentive to reduce waste in a cloud application, but requirement 4 is also an incentive to reduce waste. Since costs are a bigger driver for companies requirement 4 is more important and this requirement is nice to have as an extra incentive.

Below is a list of possible functional requirements that were considered early on, but turned out to be unsuitable. Along with the requirements is an explanation of why they are unsuitable for this research project.

- **Integration with CI/CD**
  This simply is not the best approach according to most developers.

- **Handling different types of cloud**
  All projects run their core services on the public cloud. Private cloud or on-premise is only used for very specific data storage, usually for compliance reasons, and is thus only interacted with to send/receive said information. A hybrid cloud is only used when an existing application that is already live is converted to the cloud. During this transitional phase, some aspects of the old and new systems can be needed, but in the end, only the public cloud is left. Because the public cloud is used in most cases and the other scenarios are very specific it is not useful to focus on monitoring different types of clouds for the scope of this project.

The following is a list of non-functional requirements in no particular order. All of the non-functional requirements are of equal importance and show be taken into account throughout the different phases of the design stage.

1. **Understandability** It is important to ensure the presented information, in the form of graphs and dashboards (see functional requirement 3), is easily understandable. It should require minimal to no training to understand. The information should be presented in a way to facilitate this.

2. **User-friendliness/ease of use** All to-be developed waste monitoring dashboards need to be easy to use and thus have high user-friendliness. This means the users should be able to navigate the different menus and overviews in an intuitive way

3. **Clarity of communication** Clarity of communication is very important, as misinterpreting information can cause serious problems if they result in the wrong actions being taken. Therefore it should be

clear what information each graph and table in a waste monitoring dashboard is presenting. It should also be clear to the user what the options and actions available in the menus do.

4.2 Defining Waste

In order to calculate the waste of a cloud application, waste first needs to be formally defined. During the interviews, it already became clear that the idea of waste in cloud resource provisioning could be defined in multiple ways and it is important to scope the things that are taken into consideration. One example mentioned in C18 is that it is possible to reduce waste by moving the application from X86 architecture machines to ARM machines. This would reduce the watts consumed by the application. While this can be considered a waste of power usage, the amount of time and effort required to change it is very high and it is hard to define how much waste there is when an application is using an X86 architecture. So to get a good insight into waste that can be reduced with reasonable effort these kinds of changes are left out of the scope of this research project. This means that waste will be defined based on the utilization of the resources in use, which does allow for exact metrics. Using a lot of instances that do very little is of course a much bigger waste than the previous example and is also easier and more lucrative to correct. In order to formally define the waste in resource utilization, a waste ratio is introduced, which can be used to calculate the potential savings if the waste was to be removed.

4.2.1 The Waste Ratio

Definition 4.1 (Waste Ratio)**.** *Waste is defined as the ratio of the resources that are not being utilized with respect to a preferred threshold and is in the range of* [0..100].

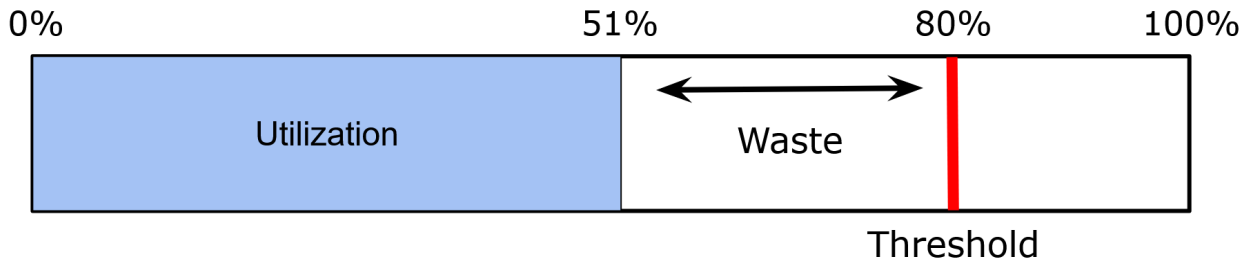


Figure 5: An illustration that shows how the total capacity of a resource can be divided up and which part of it is considered the waste ratio.

The waste ratio as defined in Definition 4.1 is always expressed in percentages. Figure 5 illustrates how the total capacity of a resource can be divided up. The blue box is an example of the current utilization of a resource. The red bar is the selected threshold for this resource and the section between the utilization and the threshold is considered waste. So in this example, the utilization is 51% and the threshold is 80%, meaning the waste ratio is 29%.

A resource can have one or more factors that give information about the utilization. For example, a compute instance has utilization metrics for CPU and memory (and possible GPU). So in order to calculate the waste ratio of a compute instance, the waste ratio for each factor has to be calculated first. The waste ratio of the factors can then be combined to get the waste ratio of the instance. By taking the average waste ratio of the instances for a certain service the waste ratio of the service as a whole can be calculated.

The threshold that is used is set to 100% by default, which means that unless the resource is being used to the full 100% there is some waste present. This is to make sure that any over-provisioning is measured

as waste if the developer has not actively interacted with the waste monitor. The threshold should be set to a different percentage based on the preferred utilization for a specific factor. The preferred utilization is different depending on the factor, the application, and the type of service. For example, a compute instance will have CPU and memory factors and it is very undesirable to have those run at 100% utilization since the users of the system will experience delays if this happens. In that case, the threshold should be set to a preferred utilization level, which is dependent on the needs and requirements of the application.

However, when the application has auto-scaling rules for scaling out in place, these rules can be used to automatically set the threshold. For example, if a compute instance has an auto-scaling rule that scales out once the instance has been running at 80% utilization or higher of the CPU for 10 minutes, the threshold will be set at 80% for the CPU waste ratio calculation. So in this example, if the CPU utilization has been at 51% for the past hour, the CPU waste ratio has been 29% for the past hour. However, if the CPU utilization would have been at 80% or over, the CPU waste would be bound to 0. This is to avoid negative values for the waste ratio and there is no waste in resources at this point, since the utilization is above the desired threshold and the auto-scaling rule will activate if this continues for 10 minutes, as was defined by the auto-scaling rule in this example. This same mechanic can be used for any other factor. Do note that in this example the threshold is determined for just the CPU waste ratio calculation, which is one of the factors in the calculation of the waste ratio of the entire instance. Having an auto-scaling rule for scaling out based on memory or GPU utilization would allow for the thresholds for those factors to be changed as well. In the absence of auto-scaling rules for scaling out, the default threshold will revert to 100%.

This works under the assumption the auto-scaling rules are set up sensibly. So instances with aggressive auto-scaling rules for scaling out should also have aggressive auto-scaling rules for scaling in. For example, if there is an auto-scaling rule that scales out if the CPU utilization is at 40% or above for 5 minutes, it is assumed this is with good cause and there is an appropriate auto-scaling rule for scaling in set up as well. This auto-scaling rule for scaling in is not used in the calculation of waste because if the rule is triggered the instance still needs some time to shut down. During this time the instance is still active and thus consuming resources that are not utilized to the preferred threshold, which means it still has waste. Once the scaling in has finished the instance will be gone and thus the waste ratio of the service as a whole will no longer take the instance into account, effectively reducing the waste ratio.

There are cases where there is intentional over-provisioning in order to deal with expected spikes in traffic. Depending on how the auto-scaling rules are set up the waste might seem a lot lower due to this. Because the over-provisioning is intentional it makes sense to correct the waste ratio for it as well, by lowering the threshold. If there is a desire to get an insight into the total waste of resources the threshold can simply be set to 100%.

The above is defined in mathematical form in Equations 3 and 4. Both equations need to be used for a single factor of an instance. So for example, the waste ratio of a CPU or the waste ratio of memory.

$$dev_i^j(\tau) = threshold_i^j - u_i^j(\tau) \tag{3}$$

$$w_i^j(\tau) = max(0, dev_i^j(\tau)) \tag{4}$$

The deviation in utilization during interval τ from the threshold for a single factor is expressed in Equation 3, where $dev_i^j(\tau)$ is the deviation in utilization during interval τ for factor j and instance i, $u_i^j(\tau)$ is the utilization of factor j for instance i during interval τ and $threshold_i^j$ is the utilization threshold for instance i and factor j. Using the deviation, the waste can be calculated for a single factor with Equation 4, where

$w_i^j(\tau)$ is the waste for factor j of instance i during interval τ. This equation is necessary because the waste should never be lower than 0.

Once the waste ratio for each factor of an instance has been calculated, the waste ratio of the entire instance can be calculated. Depending on the type of service the instance is for, the relevant factors should be selected. A compute instance has different relevant factors (CPU and memory) than a database (IOPS, compute, and storage). It is also possible that there are no utilization metrics available for a specific factor, which forces it to be excluded in the waste calculation. How much of an impact a factor should make on the waste ratio of an instance can be depended on the type of instance. In order to compensate for this weights will be used in the calculation of the waste ratio of an instance. These weights will be divided equally among the factors by default but can be changed by the developer based on the instance type. Since a memory-optimized compute instance obviously should have a bigger weight to the waste ratio of the memory factor than to the other factors, default weights can be set based on the type of instance. This allows weights to be set quickly by just selecting the instance type later on.

$$w_i(\tau) = \sum_{j=0}^{m} \alpha^j \cdot w_i^j(\tau) \tag{5}$$

$$w(\tau) = \frac{\sum_{i=0}^{n} w_i(\tau)}{n} \tag{6}$$

The waste ratio for each factor is multiplied by the weight for that factor and the result for each factor is summed. This is expressed in Equation 5, where $w_i(\tau)$ is the waste ratio for instance i during interval τ, m is the total number of factors, α^j is the weight for factor j, and $w_i^j(\tau)$ is the weight for factor j of instance i during interval τ, as calculated in Equation 4. This gives the total waste ratio of a specific instance, but to calculate the waste for an entire service the average of all the instances has to be taken. Equation 6 shows this last step in calculating the waste ratio for a service. Here $w(\tau)$ is the waste ratio of the entire service at time interval τ, n is the number of instances, and $w_i(\tau)$ is the waste ratio of instance i at τ. The waste of all instances together is the waste of the entire application.

4.2.2 Potential Savings

The waste ratio expresses the ratio of the resources for a service that is being wasted, but in order to show the importance of reducing the waste ratio, it needs to be converted to a monetary value. Since compute units are billed by type, time, and number and not by how much each instance is utilized, the calculated waste ratio of each instance can be used to calculate the amount of money wasted. However, these are just potential savings and not an exact metric on how much money can be saved. When trying to reduce waste in an application a smaller instance type will likely be used instead, but the pricing of the smaller instance does not necessarily scale linearly with the size reduction. Because the costs do not scale linearly with the instance sizes, the potential savings are only an estimate of how much money can be saved. This does give a good indication of if it is worthwhile to downscale an instance, which is very important in order to get a developer to take action.

For example, if an instance with a waste ratio of 20% is scaled-down and now only has a 5% waste ratio, this does not necessarily mean that the cost of the instance is reduced by 15%. The potential savings metric does use 15% of the cost of the instance, since the new instance type, and its hourly cost is unknown prior to downscaling.

$$c_i^{waste}(T) = C_{instance}^{hour} \cdot T \cdot w_i(T) \tag{7}$$

$$c^{waste}(T) = \sum_{i=0}^{n} c_i^{waste}(T) \tag{8}$$

Equation 7 expresses how the potential savings can be calculated, where the selected time frame T is defined as a sequence of intervals τ. Here $c_i^{waste}(T)$ is the potential savings for instance i time frame T, $C_{instance}^{hour}$ is the cost of the used instance type per hour, T is the selected time frame in hours, and $w_i(T)$ is the average waste ratio of instance i during time frame T. This gives the potential savings for instance i during time frame T. Equation 8 then takes the sum of the results from Equation 7 for all instances that were being used during time frame T, where n is the number of nodes that are being used during the selected time frame. This gives the total potential savings for a service during time frame T.

$$c^{total}(T) = C_{instance}^{hour} \cdot T \cdot n \tag{9}$$

To put this in perspective, the total costs of running the service during time frame T need to be calculated. Equation 9 expresses how to calculate this, where $c^{total}(T)$ is the total costs of the service during time frame T, $C_{instance}^{hour}$ is the cost of the used instance type per hour, T is the time frame in hours, and n is the number of instances that were being used during time frame T. This gives the total costs of the service during time frame T, which puts the potential savings in perspective and allows a developer to assess if downscaling to reduce waste is a viable option cost-wise.

$$c_i^{waste}(T) = \sum_{j=0}^{m} C_{factor}^{hour} \cdot T \cdot w_i^j(T) \tag{10}$$

In the case that the pricing of an instance is built up using the different factors, Equation 10 can be used instead of Equation 7, where $c_i^{waste}(T)$ is the potential savings for instance i during time frame T, C_{factor}^{hour} is the cost of the factor per hour, T is the time frame, and $w_i^j(T)$ is the waste ratio of factor j for instance i over time frame T. By taking the sum over the potential savings per factor the potential savings for the instance are calculated, which allows the use of Equation 8 to calculate the total potential savings for the service. An example is the billing model GCP uses for compute instances. They bill separately for the vCPU, GPU, and memory that is selected instead of using pre-defined instance types [46]. By using the waste ratio of each factor in the instance, the potential savings can be calculated much more accurately. AWS and Azure bill a virtual machine based on the instance type [47] [48], so Equation 7 has to be used.

4.2.3 Environmental impact

Based on the waste ratio the potential savings are calculated. Similarly, the potential reduction in the environmental impact of the cloud application should be calculated. Like the potential savings, this acts as an incentive for the developer and their company to reduce waste in the cloud application.

Cloud Carbon Footprint [49] is a cloud carbon emissions measurement and analysis tool, that can calculate the environmental impact of a cloud application based on cloud utilization. The tool is open source, so free to use and the code can be accessed if needed. By applying this tool the total environmental impact of a cloud

application can be calculated. The waste ratio can be used to calculate how much of the total environmental impact is caused by wasted resources. Further investigation of how to calculate the environmental impact caused by the waste in a system did not fit in the time frame of this research project. Thus creating a formal definition of the potential reduction in the environmental impact of a cloud application remains future work.

4.3 Design

In this section an architectural framework for a waste monitoring system is presented, which complies with the requirements as defined in Section 4.1. The framework will outline how a waste monitoring system should be build. This can be done by integrating with a CSP's native monitoring tool, an external tool, or a completely new dashboard. Because there is a lot of data being moved between different modules of the system, the data flow is also explained with the use of a diagram. Finally, mock-ups for a waste monitoring dashboard are presented. The information presented and how that is presented is important, but the layout of the dashboard itself is of little importance for the scope of this project.

4.3.1 Waste Monitoring Architectural Framework

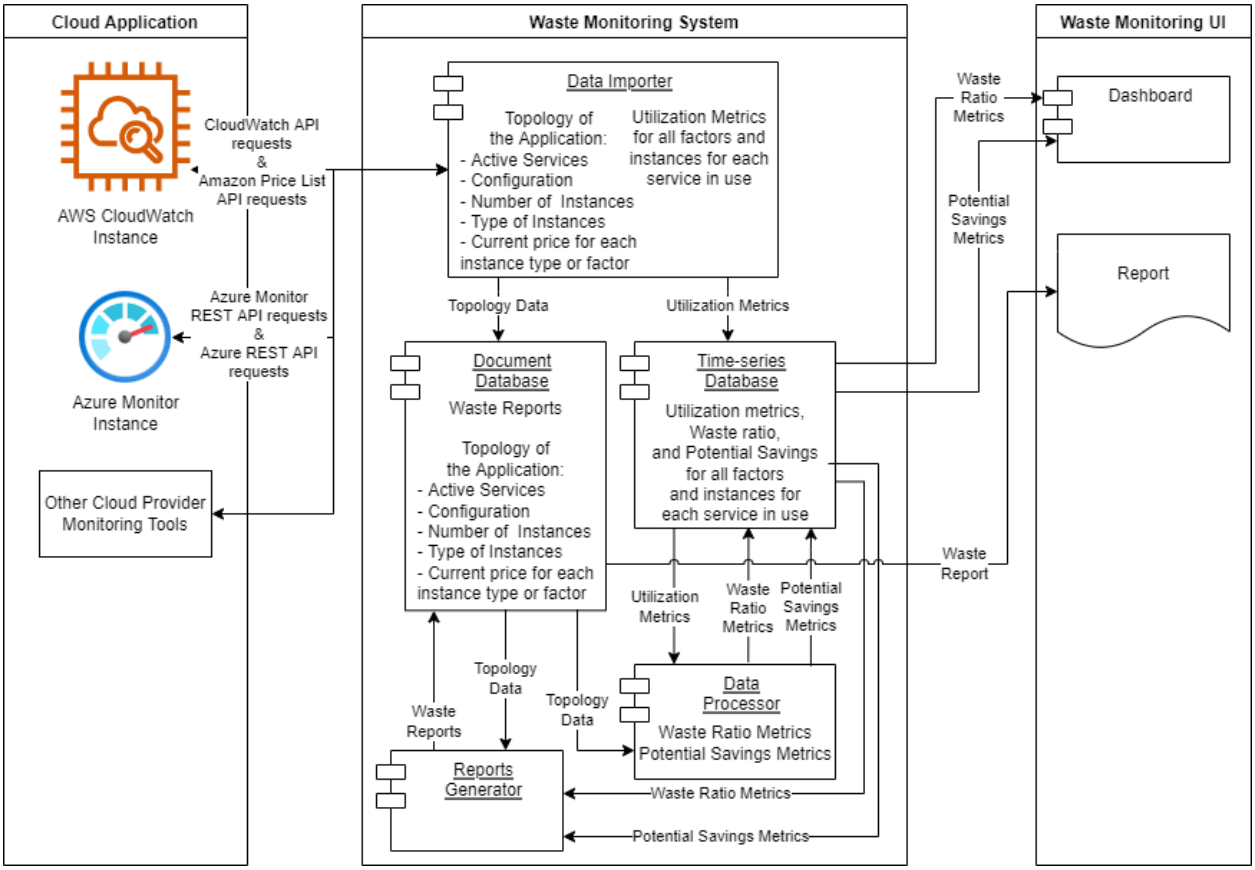


Figure 6: The architectural framework for the waste monitoring system.

The architectural framework presented in Figure 6 shows the layout for the waste monitoring system. It is split up into three different sections: The *Cloud Application* which is the existing cloud application that needs to be monitored. The *Waste Monitoring System* is what needs to be implemented to monitor waste in the cloud application. The *Waste Monitoring UI* presents the resulting waste information to a user. In the following each component will be discussed separately.

Cloud Application

The cloud application that needs to be monitored can be built using most infrastructures. It is possible to monitor waste in almost all types of infrastructure, as long as there are utilization metrics available. This includes, but is not limited to virtual machines, containers, serverless functions, and managed services. The utilization metrics of a component are needed to calculate the waste. These metrics need to be collected in a central place prior to exporting them.

On AWS a lot of services automatically collect metrics and send them to AWS CloudWatch [4]. AWS offers a lot of tools to add metrics that are not being sent to CloudWatch or to add metrics from services that do not automatically send metrics to CloudWatch. This allows for utilization metrics to be collected for VMs, containers, serverless functions, and most managed services.

Azure Monitor also automatically collects a lot of metrics, including VMs, containers, and certain managed services [5]. Azure offers the option of creating custom metrics that can be used to monitor additional resources or metrics, but it offers considerably fewer tools to help with the collection of metrics across the different types of infrastructure compared to AWS.

Other CSPs offer their own monitoring solutions and methods of collecting metrics. As long as the desired utilization metrics can be collected this should work across all platforms.

Once the metrics are collected in the cloud application itself they need to be imported into the *Waste Monitoring System*.

Topological data about the application is also needed to make sure the waste is calculated for all active services and instances. The type of instances is also needed, as well as the current price of that instance type or the price of a factor for the instance, depending on the billing model the CSP uses. This means the following information needs to be imported to the *Waste Monitoring System*: the *Active Services*, their *Configuration*, the *Number of Instances*, the *Type of instances*, and the *Current price for each instance type or factor*. The current price needs to be retrieved from the CSP itself, but the remaining topological data is available in CloudWatch or Azure Monitor. They do require different ways of retrieving the information, see Section 4.3.4 for more details.

The whole *Waste Monitoring System* can be built using the CSPs services and resources, but it can also be built using an external system or a different CSP. In the case of an external system, the information will be extracted from the CSPs native monitoring tool and imported into the external system, which costs money. The external system will calculate the waste and the potential savings. The result can be imported into the CSPs native monitoring tool or any other monitoring tool, to integrate with the monitoring tool that is already being used for the *Cloud Application*.

The other option is to leave all the data in the CSPs native monitoring tool and make use of the services available for that specific CSP. For example, if the *Cloud Application* runs on AWS all the required data is collected in AWS CloudWatch. All the compute and database components needed can be created on AWS as well, which means there is no additional cost for exporting the data to an external system. Serverless functions can be used to further reduce the cost.

Waste Monitoring System

The *Waste Monitoring System* comprises of a *Time-series Database*, a *Document Database*, and compute solutions for *Data Importing*, *Data Processing*, and *Report Generating*.

The ***Data Importer*** gets the metrics and topology data from the *Cloud Application* into the *Time-series*

Database and the *Document Database*. Based on the topological data the *Data Importer* makes sure that each factor of each instance for each active service that has utilization metrics available is properly stored in the *Time-series Database*. For some CSPs, it is possible to leave the data in their native monitoring tools and process the data from there directly, but that would limit how far into the past the data is stored. Therefore, it is better to store it in databases to retain the data as long as desired by the user.

The ***Time-series Database*** stores all the utilization metrics for the application that is being monitored. This includes all the utilization metrics available for each factor of each instance for all active services. It also stores all the waste metrics and potential savings metrics that are calculated based on the utilization metrics and topological data. The storage period for the data can be set in accordance with the preferences of the user of the system.

The ***Document Database*** stores the topological data of the *Cloud Application* over time. This consists of the *Active Services*, *Configuration*, *Number of Instances*, *Type of Instances*, and *Current price for each instance type of factor*. The *Document Database* also stores the waste reports after they have been generated by the *Reports Generator*.

The ***Data Processor*** uses the topology data from the *Document Database* and the utilization metrics from the *Time-series Database* to compute the waste as described in Section 4.2.1, as well as the potential savings as described in Section 4.2.2. This means that for each factor, instance, and service the waste and potential savings are calculated. Since the utilization metrics are over time the resulting waste metrics and potential savings metrics will be over time as well. All resulting data is written to the *Time-series Database*.

The ***Reports Generator*** uses the topology data from the *Document Database* and the waste metrics and potential savings metrics from the *Time-series Database* to generate reports over a certain time frame. The time frame can be adjusted to the preferences of the user, but every week or every month makes the most sense. The generated report is then stored in the *Document Database*.

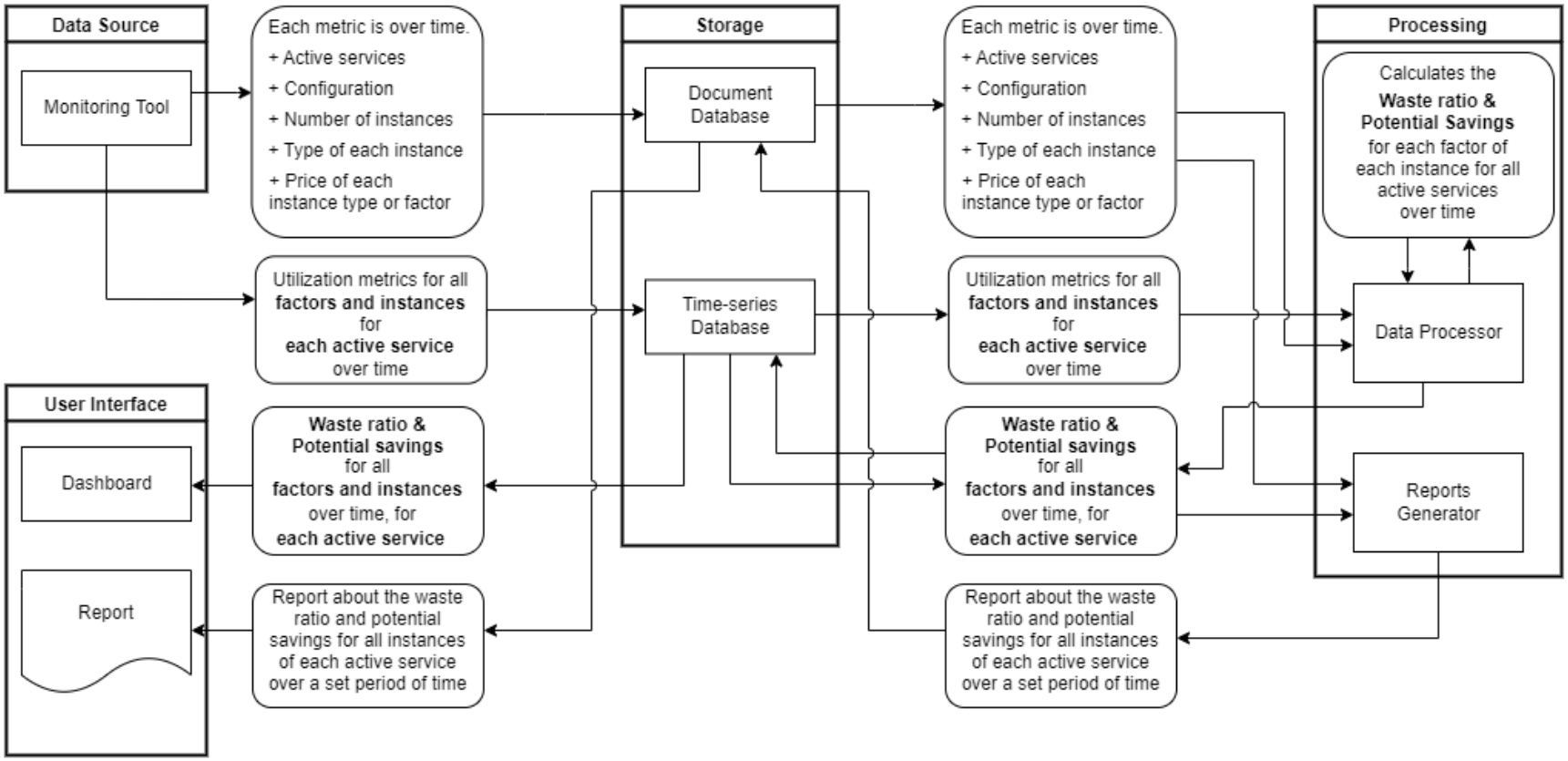


Figure 7: The data flow diagram, showing what information is going in and out of the different components.

Waste Monitoring UI

The ***Dashboard*** contains all the graphs based on the waste metrics and potential savings metrics from the *Time-series Database*. In the requirements, it is stated that the monitoring system needs to integrate with the currently used monitoring tools. This often means that the dashboard can be added to or created in the CSPs native monitoring tool, for instance, AWS CloudWatch or Azure Monitor. It can also be presented as a separate dashboard using, for instance, Grafana or Kibana. Also, see Section 4.3.3 for mock-ups of the *Waste Monitoring Dashboard*.

The ***Report*** can be displayed in any preferred environment, just like the *Dashboard*. Taking the requirement of integrating with the currently used monitoring tools into account this will likely be AWS CloudWatch or Azure Monitor, seeing as both have the ability to give alerts and reports as well.

4.3.2 Data Flows

In Figure 7 the data flow in the waste monitoring architectural framework is shown; it gives a more detailed view of what data is going in and out of each component. The *Data Source* is the same as the *Cloud Application* in the *Waste Monitoring Architectural Framework*. The same goes for *User Interface* and *Waste Monitoring UI*. *Storage* and *Processing* together form the *Waste Monitoring System* part of the *Waste Monitoring Architectural Framework*.

The *Storage* section contains the *Document Database* and *Time-series Database*. The data going out of the *Monitoring Tool* and into the *Document Database* is the topological data. This includes the *Active services*, their *Configuration*, the *Number of instances*, the *Type of each instance*, and the *Price of each instance type or factor*. This information is needed to determine for which factors, instances, and services waste should be calculated and what kind of configuration they are in. The *Current price for each instance type of factor* is needed to calculate the potential savings metrics. The data going out of the *Monitoring Tool* and into the *Time-series Database* are the utilization metrics for all factors and instances for each active service over time. This means that all the available relevant utilization metrics for the *Cloud Application* are put in the *Time-series Database*.

The *Processing* section contains the *Data Processor* and the *Reports Generator*. The *Data Processor* gets the topological data from the *Document Database* and the utilization metrics from the *Time-series Database*. Using this data the Waste ratio and the Potential savings are calculated for all factors and instances for each active service over time, which is put into the *Time-series Database*. The *Reports Generator* gets the topological data from the *Document Database* and the Waste ratio and Potential savings data from the *Time-series Database*. The data is used to generate reports over a set period of time and those reports are put in the *Document Database*.

Finally, the *User Interface* section contains the *Dashboard* that gets the *Waste ratio* and *Potential savings* data from the *Time-series Database* and the *Report* which shows the reports from the *Document Database*.

4.3.3 Waste Monitoring Dashboard Mock-ups

To give a visualization of the information that will be presented in a waste monitoring dashboard a set of mock-ups have been created. The information that is presented and how that is done is important, but the layout of these mock-ups is of little importance for the scope of this project. Note that all presented graphs and figures are just for the mock-up and do not necessarily correlate to each other correctly. There is a mock-up for the *General Overview* of the waste dashboard, the *Specific Service Overview* that contains more detail for a specific service, and the *Weights Options View* where the weights for each factor that is being used in the waste ratio calculation can be changed (see Section 4.2.1).

General Overview

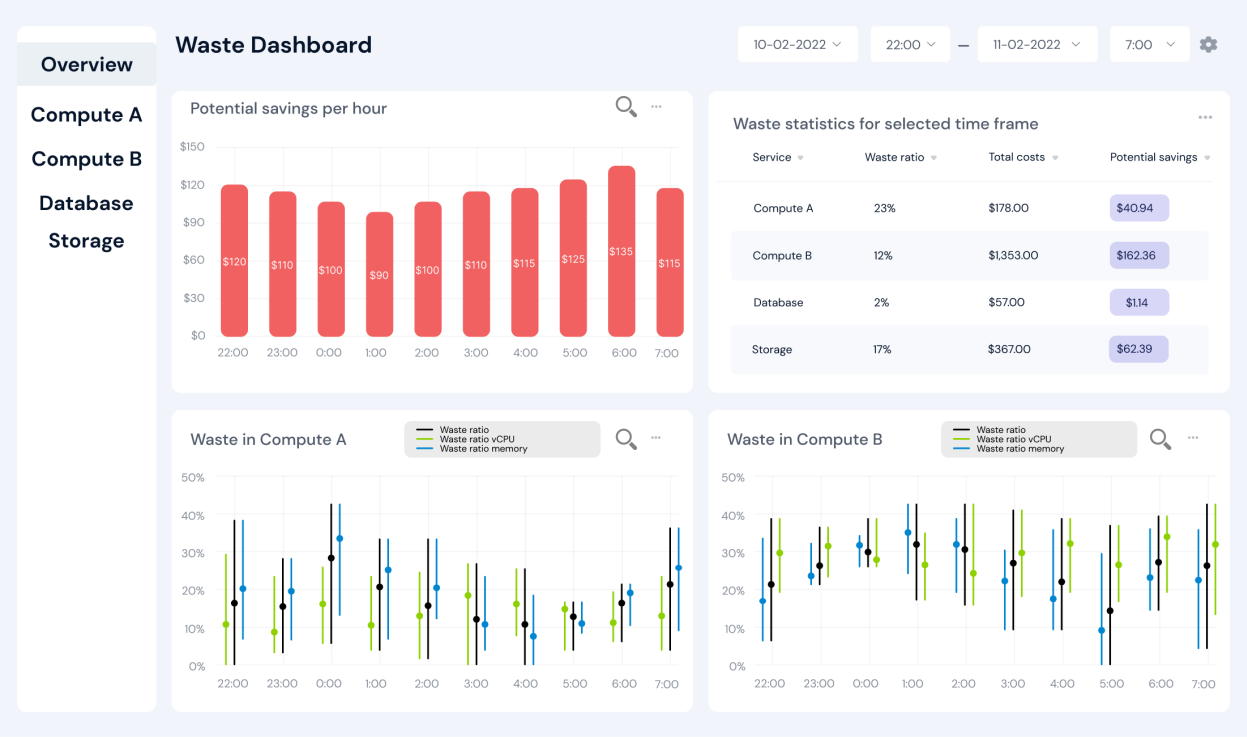


Figure 8: A mock-up of the general overview of the waste monitoring dashboard

In Figure 8 the mock-up for the general overview of the waste monitoring dashboard can be seen. This contains the information about waste for the application as a whole. On the left are buttons for the general overview and for each service that is in use. On the top right, the time frame for which the waste information should be displayed can be set. There are four boxes in the middle of the dashboard. The top left box is for the total potential savings, this shows the potential savings over time for the entire application based on the potential savings of all the active services in the selected time frame. The potential savings are calculated per hour but depending on the selected time frame the potential savings can be aggregated per multiple hours, days, weeks, or even months. By showing the potential saving over time it is easier to see if there was a change where the potential savings suddenly went up or down. Whereas this graph shows the potential savings over time, the box to the right of it shows the waste statistics over the entire time frame. The table contains the average waste ratio of each service in use during the selected time frame, the total costs of running each service for the duration of the time frame, as well as the total potential savings for each service during the time frame. This gives a comprehensive overview of the waste over the whole time frame as opposed to presenting the data over time. The two boxes at the bottom show the waste ratio for specific services in the application over time. It is possible to scroll down to show the other services that do not fit on the screen. These graphs use a min-max-median plot to show the waste ratio for the service. In black is the waste ratio for the entire service using the weights for each factor in the calculation. In green and blue are the relevant factors in the waste calculation to show more detail on how the waste is divided. If the magnifying glass icon is clicked, a more detailed overview of that specific service is shown, but it is also possible to use the buttons on the left to do this. Being able to zoom in on a certain aspect of the system was something that was considered lacking in monitoring tools according to some of the interviewees (see Section 3.4 - *Lacking in Monitoring*).

Service-specific overview

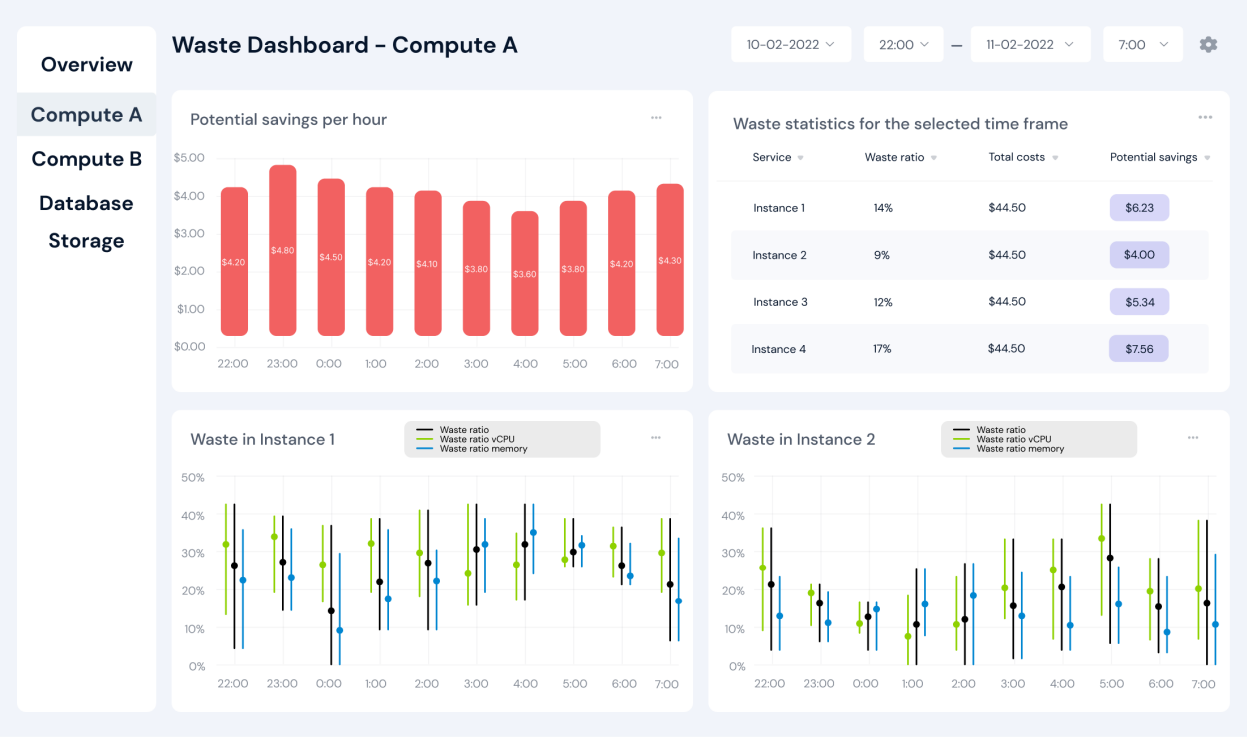


Figure 9: A mock-up of the dashboard for a specific service

In Figure 9 the mock-up for the overview of a specific service can be seen. This contains more detailed information than the general overview for a specific service, including the waste ratios for each instance. The buttons on the left and the time frame selection buttons on the top are the same as for the general overview. The box on the top right now shows the potential savings for this instance only, instead of the whole application. The information in the table in the box to the right is now for each instance within the service instead of for each service. And finally, the boxes on the bottom show the waste ratios for each specific instance. The setup is the same as for the general overview otherwise.

Weights Options View

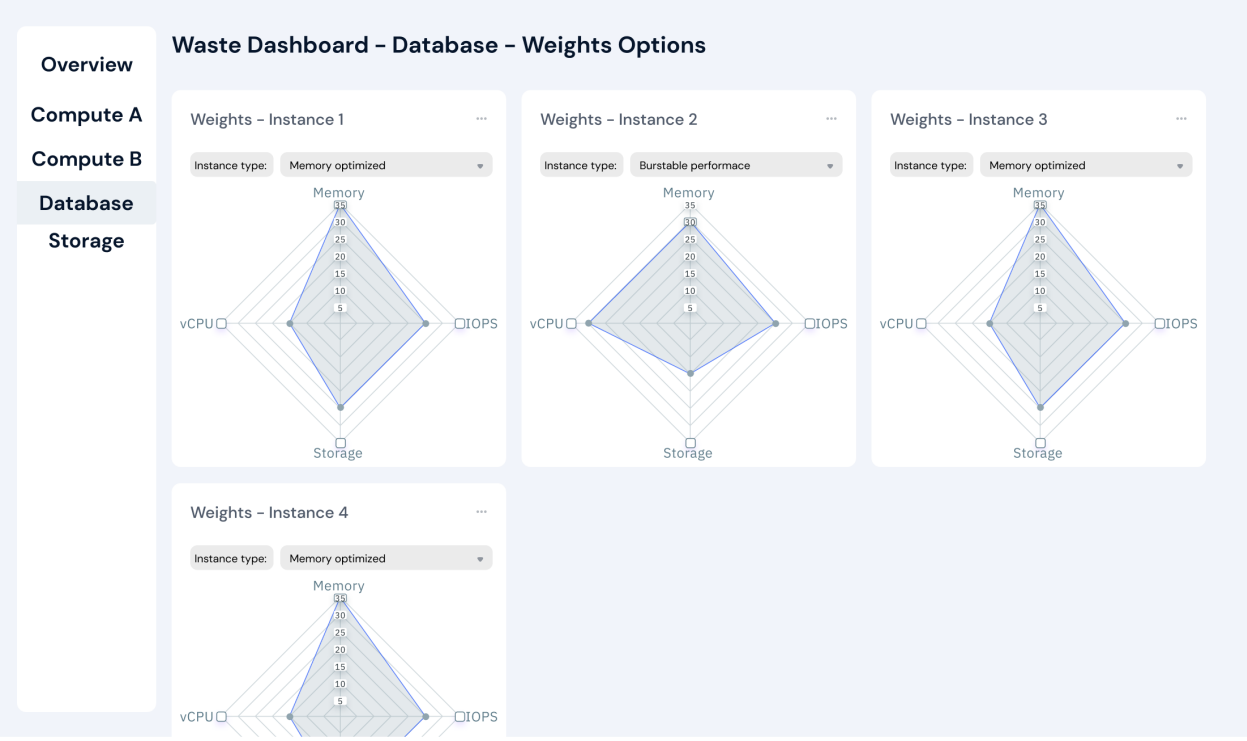


Figure 10: A mock-up of the weights options view in the waste dashboard

In Figure 10 the mock-up for changing the weights in the waste ratio calculation (see Section 4.2.1) can be changed. Depending on the service that is selected, the factors (which are vCPU, memory, IOPS, and storage in this example) for which a weight is assigned are different. Depending on the service and the instance type different default weights are used. The system provides these default weights for the most common instance types. A memory-optimized compute instance will have a 50/25/25 distribution for memory/CPU/GPU respectively by default. A CPU-optimized compute instance will have a 50/25/25 distribution for CPU/memory/GPU respectively by default. If the instance type is not selected or not a common one, the weights will be divided equally among the factors. A user can manually set up default weights for specific instance types, allowing the desired weights to be quickly applied later. This menu shows how the weights are spread out by using a radar graph but also lets the user change these weights to be more specific for the use case at hand.

4.3.4 Implementation Sketch

Functional requirement 1, *Integrate with currently used monitoring tools*, requires the waste monitoring system to be implemented for a specific monitoring tool. The design of the waste monitoring system allows the system to be fully integrated with the currently used waste monitoring tool or be created as a separate system and only import the resulting dashboard back into the currently used monitoring tool. Which waste monitoring tool it should integrate with depends on which cloud application would need to be monitored. For the scope of this research project, it was deemed unnecessary to create an actual implementation. This is because the waste monitoring system presented is applicable for any monitoring tool, implementing it for one of those would not show the universal value. The time window for the research project also did not allow for such an implementation to be made and properly tested. This is why an actual implementation of the waste monitoring system is left for future work.

To make sure that such an implementation is possible the groundwork of how to approach such an implementation has already been done for AWS and Azure. As discussed in Section 4 - Cloud Application, metrics and logs of resources are collected in the CSPs' native monitoring tools. This means that this information needs to be extracted from these monitoring tools to be put into the *Waste Monitoring System* itself. Because the *Waste Monitoring System* can be fully integrated with a monitoring tool or only put the resulting dashboard and reports back into the monitoring tool being used, there are two different ways of extracting data.

AWS

If the cloud application to be monitored is deployed on AWS, CloudWatch is used to collect all the necessary metrics and logs. The *Waste Monitoring System* can be built using AWS services and resources, using another CSP, or even using on-premise infrastructure. There are two suitable ways of getting the required data out of CloudWatch.

The first option is using Amazon Kinesis Data Firehose [50] can load real-time streams into data lakes, warehouses, and analytics services. It also allows the use of HTTP endpoints [51] which can be utilized to stream the data to various database services managed by Amazon.

The alternative is using CloudWatch API requests [52]. Metric data can be exported and imported, as well as dashboards. This means that the metric data can be put into any database and the whole waste monitoring system can be deployed to something other than AWS and the resulting dashboards or metrics can be put back into CloudWatch.

The topological data should be retrieved using CloudWatch Application Insights API requests and to get the pricing information about the current instances the AWS Price List API [53] should be used. Once the data is put into databases the processing of the data can be done using a compute resource on AWS or from anywhere else. The result is put back into the databases and that can in turn be put back into CloudWatch or any other monitoring tool that is being used.

There might be other possibilities to extract the information from CloudWatch, but the above two options show that this should be no issue when implementing the *Waste Monitoring System* if the application is running on AWS.

Azure

If the cloud application to be monitored is deployed on Azure, Azure Monitor is used to collect all the necessary metrics and logs. The *Waste Monitoring System* can be built using Azure services and resources,

using another CSP, or even using on-premise infrastructure.

The Azure Monitor REST API [54] makes it possible to use HTTP requests to extract metrics and logs, including the topological data. The data can be put into any database including the databases Azure offers. Data about the pricing of the resources should be retrieved in a similar manner using the Azure REST API [55]. The data can be processed using a compute resource on Azure or from anywhere else and the result can be put back into the databases. This data can then be put back into Azure Monitor using the REST API, or put into any other monitoring tool that should be integrated with.

Like with AWS, there might be other possibilities to extract the needed information from Azure Monitor, but this option shows that there should not be an issue when implementing the *Waste Monitoring System* if the application is running on Azure.

5 Evaluation

In this chapter, the requirements that were presented in Section 4.1 will be evaluated. The *Waste Monitoring Architectural Framework* as presented in Section 4.3.1, will be checked for the functional requirements. To check the non-functional requirements, a questionnaire has been conducted. The questionnaire itself and the results are presented next. In the last section of this chapter, the feedback from the questionnaire will be discussed and possible solutions to issues that came up are presented.

5.1 Framework

In this section, the functional requirements are evaluated based on the *Waste Monitoring Architectural Framework* and the *Waste Monitoring Dashboard mock-ups*. Note that some improvements are suggested and discussed in Sections 5.2.1 and 5.2.2 respectively.

1. **Integrate with currently used monitoring tools**
   Integrating with the currently used monitoring tool means the *Waste Monitoring Architectural Framework* needs to be suitable for basically any monitoring tool. Since AWS CloudWatch and Azure Monitor are the most popular monitoring tools amongst the interviewees, there has been a focus on integrating with them, but the way the framework is designed it should be possible to use it for any monitoring tool. The required data can be extracted from the CSPs' native monitoring tools and put into databases, as discussed in Section 4.3.1. Once the data is processed, the resulting waste ratios, potential savings metrics, and waste reports are also put in the databases. These results can be sent back to the CSPs' native monitoring tools, but the results can also be sent to any other monitoring tool, as long as the tool supports an outside data source for custom metrics. In Section 4.3.4 it is explained why it was deemed unnecessary to create an implementation of the *Waste Monitoring Architectural Framework*. The section also presents the groundwork for implementing it for AWS CloudWatch and Azure Monitor, which shows that the feasibility of the individual parts of implementation has been anecdotally checked and should not cause any problems. It is also important to note that the waste monitoring system can be implemented on the same CSPs resources and services as the cloud application to be monitored, but it can also be implemented using another CSP or on-premise resources. The waste monitoring system only needs to transfer the resulting data, dashboards, and reports to the currently used monitoring tool of choice.

2. **Waste should be monitored based on CPU and memory metrics**
   As explained in Section 4.3.1, the CSPs' native monitoring tools collect metrics and logs from the various resources and services that are being used for a cloud application. The CPU and memory metrics are among the collected metrics and are imported into the *Waste Monitoring System* to be processed. But any utilization metric can be used to calculate a waste ratio. This means that if there are other relevant factors for an instance that have utilization metrics available, they should also be imported into the *Waste Monitoring System*. The waste ratio is calculated for each factor and combined to get the waste ratio for an instance or service, as described in Section 4.2.1. Because this takes more utilization metrics into account it gives a more accurate and detailed view of the waste in a cloud application.

3. **Results should be presented in graphs and dashboards**
   In Section 4.3.3 a set of mock-ups for the waste monitoring system are presented. In Figure 8 the mock-up for the general overview of the waste dashboard is shown. This dashboard contains a graph of the potential savings over time, a table with statistics of each service in the cloud application, and graphs displaying the waste ratio over time for each active service. Figure 9 shows a similar

dashboard, but for a specific service. The potential savings over time are only for this specific service and the table with statistics shows the statistics per instance. The graphs showing the waste ratio over time are for each instance. Thus, these mock-ups show that all the results from the waste monitoring system can be presented in graphs and dashboards effectively. In Section 5.2.1 feedback about the mock-ups from the interviewees is presented. The feedback and various improvements are discussed in Section 5.2.2.

4. **The connection between reducing waste and cost savings should be made clear to users**
   In Section 4.2.2 the potential savings metric is introduced. The equations presented make it possible to calculate the potential savings based on the waste ratio of a resource. The potential savings are based on the costs of the currently used instance types of the resources. Because it is unknown which instance type the user of the waste monitoring system will choose to scale down to it is not possible to calculate a more accurate savings metric. If the new instance type would be known the price of which could be used to calculate a more accurate potential waste metric. In Section 5.2.2 the idea of predicting which instance type should be scaled down to is discussed. Another issue in the precision of the potential savings is that it is likely that the waste will not be brought down to 0 exactly when a smaller instance type is used. Because it is likely that there is a very small amount of waste left the potential savings are likely slightly off as well. But by showing the user of the waste monitoring system how much money can potentially be saved by reducing waste a clear incentive to do so is created, which is the goal of this requirement.

5. **Generate reports on a schedule or whenever there is significant waste**
   The *Waste Monitoring Architectural Framework*, as presented in Section 4.3.1, includes a *Report Generator*. This report generator is set to generate reports on a specific time interval (e.g. every week or month) or whenever there is significant waste. These reports are based on the data available in the *Document Database* and the *Time-series Database*. This means that the waste ratio over time per factor, instance, service, and the whole application is used. As well as the potential savings for each instance, service, and the whole application. Together with the topological data of the cloud application a full report of the waste in the cloud application can be created. The generated report is stored in the *Document Database* and then send to the monitoring tool which the waste monitoring system is integrated with. The user will receive the report after a set period of time or whenever significant waste is detected.

6. **It needs to be able to handle applications using various architectures and types of infrastructure**
   As discussed in Section 4.3.1 the CSPs' native monitoring tools collect information about the services and resources used for a cloud application. The most common services and resources automatically send their metrics and logs to AWS CloudWatch and Azure Monitor. AWS offers a lot of tools to add metrics and logs of other services and resources as well. Azure offers less elaborate tools to do this, but these should still allow any other resource or service to send metrics and logs to Azure Monitor. This means that all the required data for the waste monitoring system is collected in the CSPs' native monitoring tools, regardless of the architecture or the types of infrastructure used. The *Waste Monitoring System* extracts the needed data from the CSPs' native monitoring tool to process it. As long as the utilization metrics and needed topological data of a service or resource can be sent to the CSPs' native monitoring tool the *Waste Monitoring System* can handle the used architecture and types of infrastructure.

7. **[Optional] The connection between reducing waste and reducing the environmental impact should be made clear to users**
   As discussed in Section 4.2.3 the time window of this research project did not allow the creation of a metric for the potential environmental impact reduction. The potential savings metric (as discussed at

functional requirement 4) already creates an incentive for the user of the waste monitoring system to reduce waste. The potential savings metric also offers a good argument to the managers of a developer as to why the developer should invest time in reducing waste. The same is true for the potential environmental impact reduction. Because they ultimately serve the same purpose of creating an incentive to reduce waste, this one is left as future work.

5.2 Questionnaire

To verify the waste monitoring system's design, the definition of waste, and make sure all the non-functional requirements are met, the interviewees were asked to fill out a questionnaire. This questionnaire explained the various aspects of the waste ratio calculation, the calculation of the potential savings, and it presented the set of mock-ups of dashboards that visualize the resulting information (as presented in Section 4.3.3). The questionnaire as presented to the interviewees can be seen in Appendix A.

All the names of the interviewees are anonymized for privacy reasons. Also, note that Interviewee D was unavailable for the questionnaire and the *Senior Solution Software Engineer* of their team, who had supervised the interview, completed the questionnaire instead.

5.2.1 Results

In this section, the results of the questionnaire are presented. The feedback is explained, but not discussed. For the discussion see Section 5.2.2.

Waste calculation

Table 2 shows that the waste ratio as defined in Section 4.2.1 was received as very useful among the interviewees. Being able to see the waste ratio metric on a service, instance, and factor level was also received as useful. While the waste ratio metric is seen as useful in general, there are some concerns in regards to applications that have intentional over-provisioning to account for spikes in the workloads.

> *"I think this is good, and would give good insight into over provisioning of resources. Sometimes over provisioning is necessary for spiky workloads that can't be quickly scaled up / down or easily be refactored, but seeing these metrics would help make it clear the cost savings if the application was refactored."* - Interviewee F (FQ1)

> *"Seems good; with capacity planning you need to take into account some resources to be readily available (e.g., to handle spikes)"* - Interviewee C (CQ1)

A point of attention, according to Interviewee B, is to consider the timescale on which the waste ratio should be reported.

> *"It's an easy-to-use number that quickly shows how much waste is there. Point of attention would be I think to decide what the timescale is to report this waste number on."* - Interviewee B (BQ1)

Interviewee A remarked that the waste ratio metric is not useful in isolation. Other metrics like availability and performance need to be available as well in order to make the waste ratio useful.

> *"I think it can be useful to have insights into the waste as defined above, but that it also needs to be looked at from other metrics like for instance availability and performance (response time etc.). In my view this metric is not useful in isolation."* - Interviewee A (AQ1)

	A	B	C	D	E	F
How useful is the waste ratio metric as defined above in your opinion?	3	4	3,5	4	4	5
How useful is it to have the waste ratio metric available on a service, instance, and factor level? (with factor we mean a specific aspect of an instance for which the utilization can be measured, e.g. vCPU, memory, etc.)	4	3	3,5	4	4	5
How useful do you think the ability to manually change the threshold is?	4	3	4	2	4	5
How useful do you think basing the default threshold on the auto-scaling rule for scaling out is?	4	5	3	5	4	3
How useful do you think using weights for each factor is in calculating the waste ratio of a whole instance?	3	4	3	3	4	5
How useful do you think having default weights based on the service and instance type is?	4	5	3	3	4	5
How useful do you think converting the waste ratio to a monetary value is?	5	4	4	5	5	5
How useful do you think the potential savings metric is?	3	5	4	5	5	5
How understandable do you think the presented information from this view of the dashboard is?	4	4	4	4	3	5
How useful do you think the information about potential savings presented in the graph on the top left is?	2	3	4	5	3	5
How useful do you think being able to see the potential savings over time is?	4	4,5	4	5	3	5
How useful do you think the table showing the data over the entire time frame is?	4	5	4	4	3	5
How useful do you think being able to see the waste ratio per service over time is?	4	4	4	5	3	5
How useful do you think being able to get a more detailed overview for a specific service is?	4	4	4	5	4	5
How understandable do you think the information presented about specific instances is?	4	3	4	5	4	5
How useful do you think the information presented about specific instances is?	3	3	4	5	4	5
How useful do you think being able to change the weights manually using a chart is?	3	3	3	3	4	5

Table 2: The Likert-scale questions asked about the various aspects in the questionnaire with the score given by each interviewee.

Using a threshold for the waste ratio calculation was also generally well-received, as can be seen in Table 2. The interviewees provided various points of feedback about the different implications of using a threshold. The first remark is about a feature that creates a statistical model that uses the KPIs (Key Performance Indicators) and variability of the resource consumption to estimate the impact of scaling down a certain resource. This would give an insight into the performance of the application after scaling down.

> *"For an advanced (future) implementation you might be able to create a statistical model that considers the KPIs and variability of the resource consumption. Based on this data you might be able to say things as: within a 99% confidence level, you can decrease your resource size by 30% while still be able to handle all requests within the set threshold value of 300ms."* - Interviewee B (BQ2)

A concern of Interviewee D is about being able to manually change the threshold. This would allow developers to actively hide waste in a system by lowering the threshold without a valid reason.

> *"I understand being able to change it manually, but this does introduce the risk that teams manually configure it incorrectly to hide waste because they want to hide a problem that they get no time for to fix."* - Interviewee D (DQ1)

Interviewee A and Interviewee F are not in agreement with each other about the default threshold. Interviewee A states that it is undesirable to have a resource run at 100% utilization for almost any type of resource. Thus the default threshold should not be set to 100%. Interviewee F, on the other hand, argues that the threshold should be at 100% since any percent less than that is technically waste. It is important to consider that over-provisioning is necessary for some systems.

> *"The point where performance degradation is noticeable, for instance response times get above a set threshold. Might be worth considering not setting the default at 100%, because it is almost never desirable to have services run with 100% load all the time."* - Interviewee A (AQ2)

> *"I think it would be best to base it still on total provisioned vs total used. Indeed while some amount of over provisioning is necessary to leave room for growth, anything less than 100% CPU utilization is still waste and should still be captured."* - Interviewee F (FQ2)

Using the auto-scaling rule for scaling out as the default threshold was well received. But Interviewee B stated that using a threshold like this would mean the slack that is reserved for the functioning of the system is never evaluated as waste. Especially when auto-scaling rules are not configured optimally this would obfuscate potential waste. This is in line with the statement of Interviewee F about always setting the threshold at 100%.

> *"When using a threshold, you will take into account any slack available in the system that's necessary for successful operating the platform; i.e. the room necessary for scaling. However, you're also accepting waste in the system without measuring it. When the scaling rule is for instance at 60%, you're never taking the 60-100% range as waste into account. However, when scaling rules aren't configured optimally it might still be considered as waste."* - Interviewee B (BQ3)

Interviewee C references two web pages [2] [3] about *Site Reliability Engineering* (SRE). SRE is a set of principles and practices which offers a different way of working than, for instance, DevOps [56]. This means that the auto-scaling rules for scaling out might be based on different things than just the load or traffic.

Using weights to define the importance of each factor and calculate a waste ratio for an instance or service was received slightly less well, but still generally positive, according to the results in Table 2. By selecting the right weights for the currently used instance type and size the waste metric is most helpful according to some of the interviewees.

> *"This is a good idea and would help in the selection of the current instance size / type."* - Interviewee F (FQ3)

> *"Most instances I've worked with either had a CPU or memory bottleneck. So allocating appropriate weights is helpful in getting accurate waste numbers."* - Interviewee B (BQ4)

But other interviewees are concerned about the resulting figure and do not think combining the factors this way is a good idea. The waste ratio per factor is an exact number, but by combining the factors using weights

[2] Auxon Case Study: Project Background and Problem Space, `https://sre.google/sre-book/software-engineering-in-sre/`

[3] Handling Overload, `https://sre.google/sre-book/handling-overload/`

the resulting figure could be skewed.

> *"I'm not entirely sure this makes sense; I don't think the amount of CPU/MEM/GPU of a machine should be weighted when calculating an absolute figure. That would skew the outcomes since a workload that is CPU heavy will need to have more room to handle spikes."* - Interviewee C (CQ2)

> *"Per factor makes sense, but I wouldn't combine them as a weighted average."* - Interviewee D (DQ2)

Another concern in combining the factors using weights is that it might unintentionally obfuscate waste. As Interviewee A states in the quote below, when the weights are adjusted for a CPU-optimized compute unit, the waste for the memory factor gets a low weight. This might cause a relatively big waste in memory to go unnoticed, while it could be an easy fix.

> *"It might help when trying to determine priority, but what does an overall score mean? If I have large CPU utilization but have way too much memory than that might be an easier candidate to optimize than one where I have some waste CPU and some waste mem, although from what I gather they would get similar scores."* - Interviewee A (AQ3)

Potential Savings

Table 2 shows that converting the waste ratios to the monetary value of potential savings was received as extremely useful amongst the interviewees. Since costs are a very big driver in projects the potential savings metric makes waste a much more tangible problem for organizations.

> *"This is very useful for a business since profit is key."* - Interviewee D (DQ3)

> *"You need this, because historically looking at changing behavior the most immediate and effective trigger to change behavior is money."* - Interviewee E (EQ1)

> *"Great idea. This also makes it more tangible to an organization to see what the savings are. 30% waste on a t2.micro is very different to 30% waste on x1.32xlarge."* - Interviewee F (FQ4)

The accuracy of the figure is of concern though. While it is difficult to give more accurate indications of possible savings than the potential savings (as defined in Section 4.2.2), one interviewee suggested that a new instance type should be selected automatically based on the amount of waste in the current type. Using this new instance type the new costs could be calculated, giving a more accurate savings figure.

> *"I think you would need to look at what a realistic scenario is for downscaling. Just saying 10% waste means x savings doesn't make sense if there is no 10% lower compute option. Overall I do think that converting it to monetary value makes sense because you can focus on the things that can potentially get you the most savings first."* - Interviewee A (AQ4)

> *"Useful, but only if the value is (somewhat) accurate"* - Interviewee B (BQ5)

Mock-ups

The mock-up of the general overview was received as understandable and the table showing the figures for the selected time frame is considered useful. When asked how useful the information in the top left box of the mock-up of the general overview (that has a graph showing the potential savings over time, see Figure 8) was the interviewees gave a relatively bad score (see Table 2). But when asked how useful being able to see

potential savings over time is, the interviewees gave it a relatively very high score. Being able to see the waste ratio per service over time is also considered very useful.

Something that interviewees missed in the presented mock-up was what the impact of scaling down would be. Predicting the impact of downscaling on a system was suggested by two interviewees. Having some indication about the risk of failures is very valuable when deciding if downscaling to reduce waste is a viable option. Aligning with this is the concern about how to use the information to make an actionable decision.

> *"If I'd operate a platform, I would like to know what the impact would be if I would downscale resources. Based on historical traffic and resource consumption you could model the impact and calculate a confidence score. That would be interesting. For instance, if I would have 50% unallocated capacity, with what percentage could I downscale this resource without running a risk if system outages/failures?"* - Interviewee B (BQ6)

> *"The overall information is presented well. I do want to highlight that translating this information into actionable/actual savings is a lot trickier. E.g., I would like to know whether actually downscaling makes sense based on historical data (e.g. will the downscale not result in overload in the future damaging my business)."* - Interviewee C (CQ3)

> *"I think it sums up the information nicely, but I wonder how actionable these figures are. How would someone be able to really decide, okay this instance can be downscaled."* - Interviewee A (AQ5)

While the information presented in the general overview is well received, it was considered to be quite a lot by Interviewee D. Having less information on the screen at once and being able to zoom into specific things is preferred. Setting the time frame based on a date and time is also seen as overly complicated and showing a week, month, quarter, or year would be better. Interviewee B agrees that the time frame should be based on weeks, months, or years.

> *"For an overview, this is quite a lot of data. I would rather see bigger numbers on the overall savings, that you can then drill down into if needed. And instead of complicated date filtering I would just show week, month, quarter, year as default. Keep it simple."* - Interviewee D (DQ4)

> *"Time scales per hour are not as relevant I think. More useful is per day/week/month."* - Interviewee B (BQ7)

Another concern is regarding the speed at which an application can be scaled up or down while reaping the benefits of reducing waste. This is however not a problem to get the conversation started about reducing waste based on the information presented on the dashboard.

> *"I really like having this data laid out. The one issue it raises is that a lot of organisations are not able to scale their applications up/down by the hour to reap the benefits / cost savings, but this should help start the conversations about what money could be saved if applications and workloads are refactored."* - Interviewee F (FQ5)

The second mock-up, showing the more detailed view for a certain service (see Figure 9), was also received as understandable and useful.

> *"Yep this is also good, and important to have this breakdown. It is common for organizations to run multiple workloads with varying levels of CPU / GPU intensity (e.g. for machine learning), so being able to track it on service / departmental view is important."* - Interviewee F (FQ6)

> *"Per service is quite important for reducing costs."* - Interviewee D (DQ5)

One of the interviewees suggested that it would be handy to have the selected threshold displayed with the graphs, so it is clear what the numbers are based on.

> *"I think it would be handy to show the threshold % used per instance so you know what the waste numbers are based on. Other than that I think it is a clear overview."* - Interviewee A (AQ6)

Interviewee E stated that they are missing guidance on how to tackle reducing the waste, which others have mentioned during the questions about the general overview as well.

> *"This looks good for a specific service. The only thing I am missing is a clear trigger to action. What does someone need to do to solve that waste? Any "others have done x to reduce this type of waste by x percent""* - Interviewee E (EQ2)

The last mock-up (see Figure 10) was received as considerably less useful than the other mock-ups. This is in correspondence with the concerns about using weights per factor to create a waste ratio for the instance or service as a whole. One interviewee also stated that users should be well informed of the consequences of changing the weights.

> *"I think it looks nice the way it is presented, my question would still be as previously mentioned, what is the use of the weighted average waste percentage as a value in itself. Why not for instance state per instance the largest waste %."* - Interviewee A (AQ7)

> *"As mentioned above determining these weights is tricky."* - Interviewee C (CQ4)

> *"Could be useful. However, when having users change weights themselves they should be well informed of the consequences."* - Interviewee B (BQ8)

Other Remarks

At the end of the questionnaire, the interviewees were asked if they had anything else to add. Interviewee A asked about waste in storage, where files can be moved from hot storage to cold storage if they are not accessed frequently. If they are left in hot storage this will cost more and can thus be considered waste as well.

> *"I was wondering if you also thought about other waste factors like for instance storage. For instance when files are kept in a hot storage tier but are never accessed for a long time and could be better put into archive storage. I think that the potential waste there would be quite large."* - Interviewee A (AQ8)

Interviewee F gave positive feedback and is very happy with the direction the project is going. Especially being able to use the information from the waste monitoring dashboard to get organizations to see that reducing waste is, in fact, necessary.

> *"This is great stuff, and I really like the direction this is going in of being able to capture this information and make it presentable to organisations. This would also help capture one of the big areas of cloud waste, which is leaving cloud resources running when they are no longer needed. E.g. when you provision a service and then forgot to shut it down. In any organization using cloud there will be some level of cloud waste, unless they are 100% serverless and running the absolute bare minimum of what they need. Very few organizations run this way, but this data could help them start to move towards that, or at the very least reduce their waste and lower their bills."* - Interviewee F (FQ7)

5.2.2 Discussion

The results from the questionnaire (as presented in Section 5.2.1) brought up various suggestions and concerns. In this section, the concerns about using a default threshold will first be discussed and the *Cost of Flexibility* will be proposed as a possible solution. Next, the concerns about using weights in the waste ratio calculation will be addressed. After that, various other remarks and concerns will be discussed.

Default threshold and the cost of flexibility

The first big point of discussion is the use of a default threshold because interviewees are not in agreement about it. In Quote AQ2 Interviewee A argues that the default threshold should never be 100% since there are almost no services where it is desirable to have 100% utilization. In Quote FQ2, on the other hand, Interviewee F argues that the default threshold should always be 100% since that gives the actual waste in the system. It should be noted that a part of that waste is necessary for scaling purposes or spikes in the workload. Interviewee B addresses this issue exactly in Quote BQ3.

> *"When using a threshold, you will take into account any slack available in the system that's necessary for successful operating the platform; i.e. the room necessary for scaling. However, you're also accepting waste in the system without measuring it. When the scaling rule is for instance at 60%, you're never taking the 60-100% range as waste into account. However, when scaling rules aren't configured optimally it might still be considered as waste."* - Interviewee B

What this comes down to is whether or not the utilization that is higher than the threshold should be considered waste. The waste ratio is 0 if the threshold is passed, so the costs of anything above the threshold are not taken into account in the possible savings calculation. This is not possible because the threshold is put in place to leave room for the system to scale out without serving time-outs or errors to the end-users. Anything above the threshold will not be used most of the time but is needed for this purpose. Because of this, it would not be logical to consider it as part of the waste ratio (as defined in Section 4.2.1).

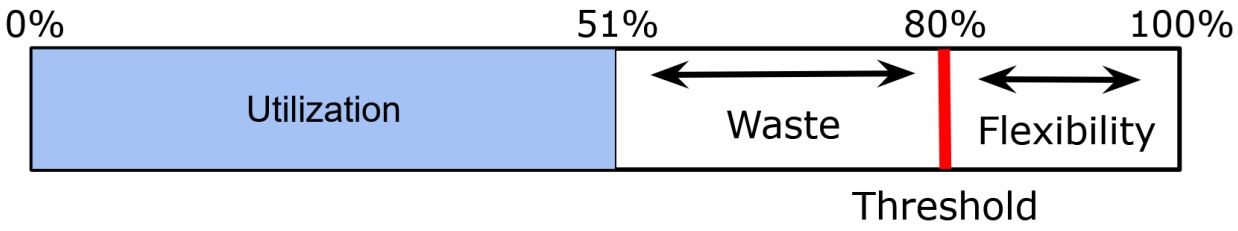


Figure 11: An illustration that shows how the total capacity of a resource can be divided up and which part of it is the flexibility that is needed for scaling purposes.

A possible solution to this is to introduce the *cost of flexibility*. Figure 11 illustrates how the total capacity of a resource can be divided up and which part of that is the flexibility needed for scaling purposes. Calculating the costs of this flexibility would give a monetary value to the section of the total utilization that is reserved for scaling purposes. The cost of flexibility allows the users to see the costs of selecting a certain threshold. For example, if the threshold is set to 80%, the 20% that is between the maximal utilization and the threshold is causing costs that are not taken into account in the possible savings calculation. Based on this the costs of keeping this flexibility in the system can be calculated.

$$flexibility_i^j(T) = 100\% - threshold_i^j(T) \tag{11}$$

$$flexibility_i(T) = \sum_{j=0}^{m} \alpha^j \cdot flexibility_i^j(T) \tag{12}$$

$$c_i^{flexibility}(T) = C_{instance}^{hour} \cdot T \cdot flexibility_i(T) \tag{13}$$

$$c_i^{flexibility}(T) = \sum_{j=0}^{m} C_{factor}^{hour} \cdot T \cdot flexibility_i^j(T) \tag{14}$$

$$c^{flexibility}(T) = \sum_{i=0}^{n} c_i^{flexibility}(T) \tag{15}$$

Equation 11 expresses how to calculate the flexibility ratio for instance i and factor j over time frame T, where $threshold_i^j(T)$ is the threshold for instance i and factor j that was used during time frame T. Using the result of this equation the flexibility for the entire instance i over time frame T can be calculated with Equation 12, where m is the number of factors, and α^j is the weight of factor j. Equation 13 shows how to calculate the costs of having this flexibility in the provided resources, where $C_{instance}^{hour}$ is the cost of the used instance per hour, T is the time frame in hours, and $flexibility_i(T)$ is the flexibility ratio for instance i during time frame T, that was calculated using Equation 12. Alternatively, Equation 14 can be used in the case that the billing model of the CSP uses a price per factor instead of the instance type. This equation should be used instead of Equation 13 if that is the case. Here C_{factor}^{hour} is the cost of the relevant factor per hour, m is the number of factors, T is the time frame, and $flexibility_i^j(T)$ is the flexibility ratio for instance i and factor j during time frame T. Finally, Equation 15 takes the sum of the costs for flexibility of all the instances in a service, where n is the number of instances used during time frame T, to get the total costs of the flexibility of the service during the selected time frame. The cost of flexibility for each service can be summed up to get the cost of flexibility for the application as a whole.

While the cost of flexibility does not map the potential waste above the threshold, it might help the user reconsider if the right threshold is used. It also does not change the fact that anything above the threshold is not considered waste in the scope of this research project. So one can still argue that the default threshold should be 100% and the room for flexibility needs to be considered when looking at the waste ratios. However, this is a good compromise between the points made by Interviewees A and F and it is in line with what Interviewee B said.

In Quotes FQ1 and CQ1, interviewees state that some systems need some over-provisioning for spikes in workloads or if a system can not scale up and down very fast. This is similar to the flexibility that is discussed above. Using the cost of flexibility the costs of intentional over-provisioning can be calculated, allowing users to see the actual costs of the intentional over-provisioning. This does not show however if the over-provisioning is of an appropriate amount, which would be incredibly hard to determine, since any time the utilization goes above the threshold, a spike in traffic is encountered and the system needs time to scale out. These spikes are likely to be irregular in time and height, thus making it hard to determine if the right amount of over-provisioning is used. Being able to see the costs of the room for flexibility might have a user reconsider if the right amount of over-provisioning is used for their specific application.

Using weights in the waste ratio calculation

The interviewees are quite divided in opinion about the use of weights in the calculation of the waste ratio. Interviewees F and B are in favor of using weights (see Quotes FQ3 and BQ4), but Interviewees C and D

are not (see Quotes CQ2 and DQ2) and Interviewee A also has their concerns (see Quotes AQ3 and AQ7). The Interviewees that are not in favor of using weights argue that it would skew the resulting waste ratio. If a compute instance is CPU optimized it would also need more room to handle spikes and thus should not have a bigger weight applied to it according to Interviewee C. Leaving flexibility in the system to account for spikes and scaling is discussed in the paragraph above and since each factor can have a different threshold the need flexibility should not matter in the resulting waste ratio. This is of course assuming that the threshold is set up properly. On the other hand are the interviewees that think the weights are a good idea because it allows the user to put more emphasis on the most important factors. Interviewee B is in favor of using weights in Quote BQ4 but later stresses that users need to be made aware of the consequences of changing the weights in Quote BQ8. This is an important point since changing the weights has a very big impact on the resulting waste ratio for the instance or service. Interviewee C also states that determining the right weights could prove tricky in Quote CQ4. Using default weights for certain types of instances can help the user to use appropriate weights, as stated in Section 4.2.1. There are two options for getting the needed information to decide on the use of weights in the waste ratio calculation. The first option is conducting an A/B experiment. For this a group of practitioners that have not read about the waste ratio yet is split into two, one group is asked to give feedback on the waste ratio calculation using weights and the other is asked to give feedback on the waste ratio without using weights. This way a non-biased opinion about the waste ratio calculation with and without weights can be compared. The second option is using a much larger sample of practitioners and asking them if they think weights should be used in the waste ratio calculation. Because the sample is much larger the results are quantitative instead of qualitative and a decision about using weights can be made.

Another concern about using weights is voiced by Interviewee A in Quote AQ3, where they state that using weights might obfuscate waste. For example, if a compute instance is CPU optimized the weights should be shifted to the CPU and not the memory. The memory might have a large waste ratio that could easily be fixed, which would go unnoticed since it has a lower weight. Using weights, the waste ratio of the instance, as described, would have a similar waste ratio as an instance that has about equally distributed waste and equal weights. So even though a specific factor is most important in an instance, the other factors might have waste that is easy to fix. In the mock-up for the service-specific view (see Figure 9) the graph for each instance shows the combined waste ratio in black, but also shows the waste ratio of each individual factor in different colors. This shows the unweighted waste ratio per factor, which allows the user to see if there are any abnormalities per factor. While the weights might obfuscate some waste in the resulting waste ratio for a whole instance, by zooming in and looking at the graph for a specific instance the user can still see the underlying data. In Quote AQ7 Interviewee A suggests showing the factor with the largest waste ratio for each instance instead of the combined waste ratio as described in Section 4.2. While this would not show all the information available this might help the user to identify waste that can be reduced more easily. An option to switch between the view containing the combined waste ratio and a view only showing the factor with the highest waste ratio can be created in the existing waste monitoring dashboard.

Predict the impact of downscaling

A point that came up a lot in the questionnaire was regarding taking action based on the information in the waste monitoring dashboard. The dashboard provides information about waste and potential savings, but what kind of actions should be taken are for the user to decide. Interviewee A wonders how the user would be able to decide what action to take in Quote AQ5. Interviewee E has a similar remark about this in Quote EQ2 but also mentions that it might be helpful to have some advice based on the actions of others. The main concern with this is that the interviewees want to know what the impact of downscaling would have on the system. Interviewee B states in Quote BQ6 that they want to know how much of the waste

can be reduced without running the risk of outages or failures. Both Interviewees B and C suggest that a prediction of the impact on the system when downscaling can be made. Interviewee C suggests using historical data for this prediction in Quote CQ3. While Interviewee B suggests creating a statistical model that takes KPIs (Key Performance Indicators) and the variability of the resource consumption into account in Quote BQ2. The option of using historical data would require previous downscaling in the cloud application. This might not always be available for a system and can thus prove hard to apply for cloud applications that have not matured yet. The option of creating a statistical model would not need this information and bases the prediction on the KPIs and the variability of the resource consumption of the application instead. This would mean that the prediction might be less accurate since it is hard to accurately model the real situation based on conditions set for the system. Both approaches are very promising, but it does not fit within the duration of this project to further research the possibilities in this direction and remains as future work.

Another issue in regard to downscaling is not knowing what type of instance would be best suited for the instance based on the current waste, which is something that is left up to the user to decide. In Quote AQ4 Interviewee A states that the possible savings metric makes no sense if there is no instance type that fits the ideal scenario for an instance's utilization. Building upon the idea of making predictions about the impact on the system when scaling down, a possible solution to this could be to also suggest which instance type should be used instead. The cost of using the new instance type can be compared to the cost of the current instance type, which would allow for a much more accurate calculation of the possible savings metric. However, selecting the right instance type for the application could prove difficult, especially because the available instance types change over time. Further research in the possibilities of estimating which instance type is best suited based on the current waste ratio needs to be done.

The accuracy of the potential savings is something Interviewee B is also concerned about in Quote BQ5 and it could possibly be improved. However, the metric is considered very useful to get companies to actually take action since costs are a big driver as stated in Quotes DQ3, EQ1, and FQ4. As discussed above, determining the new instance type automatically would allow for a more accurate potential savings metric. But the current way the potential savings are defined should still help developers in showing companies that it is important to reduce waste.

In Quote AQ1 Interviewee A states that the waste ratio is not useful in isolation and metrics like performance and availability also need to be taken into account. Looking at the application level this makes sense but looking at the system level the waste ratio and the potential savings should provide adequate information to see the amount of waste in the system. Since the waste monitoring system should be integrated with the currently used monitoring tool these additional metrics should be available to the user. Of course, integrating those metrics into the waste monitoring dashboard could help the user in making a decision about downscaling. The paragraph above about predicting the impact on the system when downscaling might also take away these concerns since these other metrics would need to be taken into consideration in order to make an accurate prediction.

Other points of discussion

Interviewee D mentions in Quote DQ4 that the mock-up of the overview of the waste monitoring dashboard has quite a lot of information in it. According to Interviewee D, it would be better to only present the most important information and allow the user to zoom in on details that are of interest. While the mock-up was created with the option of zooming in on the details of certain aspects in mind, this is of course a good modification. Allowing the users to get a quick insight into how the cloud application is doing in keeping waste to a minimum and being able to get additional information if the application has significant waste. Interviewee D also thinks the time frame selection buttons are too specific. It would be better to select

certain weeks, months, quarters, or even years, rather than being able to set the specific date and time for the start and end of the time frame. This makes a lot of sense and would help the user to select the relevant time frame much faster. The feature of selecting the time frame to the minute could be left in as an advanced feature behind an extra button. In Quote AQ6, Interviewee A suggested adding the selected threshold to every graph. Which threshold is selected makes a very large impact on the waste ratio, since the waste ratio is calculated based on the selected threshold. Displaying the used threshold thus makes it possible to get a much more accurate and complete reading of the reality from a graph and should indeed be added.

Interviewee D is concerned that allowing the user to manually set the threshold would allow them to hide waste (see Quote DQ1). By setting the threshold close to the actual utilization percentage of an instance the waste ratio can artificially be kept close to 0%. While the user of the system is assumed to use the system as intended to improve the system they are building, this is a realistic possibility. As discussed in the paragraph above, displaying the used threshold with every graph in the dashboard would give a more realistic and complete reading of the reality in each graph. Adding the threshold to each graph would also remove the possibility of hiding waste to an extent. The selected threshold is shown and it should be possible to argue why that specific threshold was selected. If a weird threshold is not questioned, it is of course still possible to hide waste.

Something odd in the questionnaire results is that the interviewees gave a relatively high score to how useful being able to see potential savings over time is, but gave a relatively low score to the usefulness of the information provided in the graph actually showing the potential savings over time. This implies that the way the information is presented in the graph is not clear enough. Figure 8 shows the graph for potential savings over time in the top left box. In the mock-up, the time unit is hours, which is too short a period of time as stated in Quotes DQ4 and BQ7. This might be a reason why the interviewees consider the information in the graph to be less useful. Another possibility is that the bar chart that has been used is not showing the information in a way that is easy to understand. Using a different kind of chart or plot might make it easier for the user of the waste monitoring dashboard to understand what the graph is representing.

In Quote AQ8 Interviewee A talks about waste factors in file storage. If a file is in hot storage but not frequently accessed it would be better to move the file to cold (or archive) storage. While this could be considered waste since hot storage is more expensive, this does not fit the waste definition as defined in Definition 4.1. Monitoring if files should be moved to less frequent access storage would be a very different project and is thus outside of the scope of this project.

Non-functional requirements

While the interviewees gave good scores to the *understandability* of the information presented in the mock-ups (see Section 4.3.3), various points of improvement have been discussed in this section. The use of a default threshold had the interviewees divided in opinions with valid arguments on both sides. The *cost of flexibility* has been proposed as a possible remedy to the concerns of the interviewees. Using weights for calculating the waste ratio of an instance was also a big topic of discussion. Even though there is no clear solution various possible improvements have been discussed.

The *clarity of communication* requirement also had mixed feedback. For example, the interviewees think it is important to see the potential savings over time, but gave a relatively low score to the graph actually showing the potential savings over time. Changing the weights for the waste ratio calculation for an instance also needs a warning to stress the consequences to the users. An example of a good point in clarity of communication is the table with the information about the application or service for the whole time frame selected. This allows users to quickly see the important information without having to read a graph.

As for the *user-friendliness*, various aspects are well received, for example, being able to zoom in on a service and seeing detailed information. But some are less easy to use than intended, like the way the time frame is selected. In the mock-up as presented in Figure 8, the time frame needs to be selected by setting an exact time and date for the start and end. This was considered unnecessarily specific and being able to select specific weeks, months, quarters, or years would be much easier.

6 Conclusion

This chapter concludes this manuscript. First, a summary of the chapters is to be given and the research questions answered. After that, the future work will be discussed.

6.1 Summary

The needed background information is provided and related work is discussed in Chapter 2. There is not a lot of literature about waste monitoring available, aside from some prior works [12] [13]. The cost monitoring systems Costradamus [14] and CostHat [15] touch upon the subject of waste monitoring but do not go into much detail. The focus of monitoring systems in literature is on monitoring VMs. The monitoring architectures, communication models, and types of resources being monitored are discussed. State-of-the-art features, like monitoring systems without probes and monitoring systems that check, predict, and take action on SLA violations, are also discussed. Lastly, this chapter discusses a selection of current monitoring and dashboarding tools as related technologies.

Chapter 3 explains how formal interviews have been conducted in order to get a deeper insight into how developers use cloud monitoring tools in practice. These interviews have been analyzed using the constant comparison method, meaning that quotes have been extracted from the transcribed interviews and codes have been assigned to them. The findings have been organized into a hierarchy of codes and presented for each of the different codes. The big differences between the literature and the findings are the type of infrastructure being used, the architecture of monitoring tools, and the integration of monitoring systems with currently used monitoring tools. The findings from the interviews presented a valuable insight into the commercial field of cloud monitoring and also provided the needed information in order to design the *Waste Monitoring System*.

The *Waste Monitoring System* is presented in Chapter 4. Based on the findings from the interviews a set of functional and non-functional requirements have been formulated for this system. Next, waste is formally defined by Definition 4.1 and equations are presented on how to calculate this. To translate the waste ratio to a monetary value the potential savings metric is presented in Section 4.2.2. Translating the waste ratio to an environmental impact is shortly discussed as well. The *Waste Monitoring Architectural Framework* is introduced after that. This framework outlines how a waste monitoring system should be built. The system can be built as a separate tool or integrated with an existing monitoring tool. Lastly, this chapter presents a set of mock-ups showing how the data can be presented to the user.

In Chapter 5 the work is evaluated. The framework is evaluated based on the functional requirements from Section 4.1. To evaluate the waste ratio and the non-functional requirements a questionnaire has been conducted among the interviewees. Based on the feedback of the interviewees' various points of improvement have been discussed, presenting possible solutions where possible and outlining how to approach further research otherwise.

6.2 Findings

The main research question Q1 is *"How to monitor and calculate resource waste in cloud-based applications?"*. The *"How to monitor"* part is answered by the *Waste Monitoring System* in Chapter 4. This requires the CSPs' native monitoring tools to collect all the needed metric and topological data of the resources and services used and it needs to be possible to extract that data. This is the case for AWS and Azure as shown in Section 4.3.4. The *"How to calculate"* part is answered by introducing the *Waste Ratio* in Section 4.2.1. The waste ratio uses a threshold to reserve part of a resource as slack for scaling. The interviewees are divided about the use of this threshold and to solve the issue the *Cost of Flexibility* is introduced

in Section 5.2.2. The waste ratio uses weights for each factor to get the waste ratio of an instance. The interviewees are in disagreement about using these weights in the calculation as discussed in Section 5.2.2. The first sub-question SQ1, *"What is the most efficient and accurate way to compute the waste in provisioned resources for a cloud application?"*, is answered by the waste ratio and the potential savings metric. While the waste ratio is the most accurate way to compute waste, the potential savings metric creates an incentive to actually reduce the waste. The potential savings are based on the price of the current type of instance because the new type of instance is unknown. This might cause deviations between the potential savings and the actual savings as discussed in Section 5.2.2. The second sub-question SQ2 is *"How can the waste be best represented to the user to get better insights?"* which is answered with the mock-ups as presented in Section 4.3.3 and various improvements are discussed in Section 5.2.2

6.3 Future Work

As mentioned before, several points of improvement were left as future work because they did not fit in the time frame of this research project. These open topics are discussed below.

The time frame of this research project did not allow the 7^{th} functional requirement, *"[Optional] The connection between reducing waste and reducing the environmental impact should be made clear to users"*, to be fulfilled. But the *Waste Monitoring System* can be easily adjusted to handle this as well. Any additional data needed can be imported into the *Waste Monitoring System* similarly to the data that is being imported for the waste ratio and the potential savings metric. Once a formal definition of the potential environmental impact reduction has been created, the *Data Processor* in the *Waste Monitoring System* can be adjusted to calculate this along with the other metrics. The result can be stored in the time-series database and can be included in the dashboards. This means that creating a formal definition of the potential environmental impact reduction and adjusting the *Waste Monitoring System* to handle it, remains future work.

The logical next step in this research is to implement the presented architectural framework. This was not possible in the time frame of this research project. The information gained from the formal interviews has made it possible to formulate requirements for a waste monitoring system and formally define waste (see Definition 4.1). Based on this the architectural framework for a waste monitoring system was designed, which fits the preferences and needs of experts in the domain of cloud engineering. The feasibility of the individual parts of implementation has been informally checked (see Section 5.1) and should cause no problems during the actual implementation.

One of the points that came up during the discussion was adding a feature to the waste monitoring system that can predict the impact of downscaling for the application. One interviewee suggested making the prediction based on historical data and another interviewee suggested building a statistical model based on the KPIs and the variability of the resource consumption. Both suggestions have potential and further research is needed to create this feature. Automatically suggesting which instance type should be used when downscaling is another feature that needs to be researched. These features make it much easier for a user of the waste monitoring system to take action to reduce waste.

The interviewees are also not in agreement about the use of weights in the waste ratio calculation for an instance. To solve this issue a larger sample of developers is needed to do a quantitative study. An alternative is to conduct A/B testing to find out what developers prefer.

References

[1] C. B. Mahantesh N. Birje, "Cloud monitoring system: A review," *International Journal of Engineering Sciences and Management - A Multidisciplinary Publication of VTU*, vol. 1, no. 1, pp. 49–55, 2019. [Online]. Available: https://www.usenix.org/conference/cooldc16/workshop-program/presentation/turk

[2] G. Aceto, A. Botta, W. de Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128613001084

[3] H. J. Syed, A. Gani, R. W. Ahmad, M. K. Khan, and A. I. A. Ahmed, "Cloud monitoring: A review, taxonomy, and open research issues," *Journal of Network and Computer Applications*, vol. 98, pp. 11–26, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804517302783

[4] AWS Cloud Watch. Accessed: 14-6-2022. [Online]. Available: https://aws.amazon.com/cloudwatch/features/

[5] Azure Monitor. Accessed: 14-6-2022. [Online]. Available: https://docs.microsoft.com/en-us/azure/azure-monitor/overview

[6] GCP Cloud Monitoring. Accessed: 24-6-2022. [Online]. Available: https://cloud.google.com/monitoring

[7] V. Andrikopoulos, T. Binz, F. Leymann, and S. Strauch, "How to adapt applications for the cloud environment." *Computing*, vol. 95, p. 493–535, 2013.

[8] L. Wei, C. Foh, B. He, and J. Cai, "Towards efficient resource allocation for heterogeneous workloads in iaas clouds," *IEEE Transactions on Cloud Computing*, vol. 6, pp. 1–1, 01 2015.

[9] Y. Liu, X. Wei, J. Xiao, Z. Liu, Y. Xu, and Y. Tian, "Energy consumption and emission mitigation prediction based on data center traffic and pue for global data centers," *Global Energy Interconnection*, vol. 3, no. 3, pp. 272–282, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2096511720300761

[10] AWS Lambda changes duration billing granularity from 100ms down to 1ms. Accessed: 14-6-2022. [Online]. Available: https://aws.amazon.com/about-aws/whats-new/2020/12/aws-lambda-changes-duration-billing-granularity-from-100ms-to-1ms/

[11] Azure Function Pricing. Accessed: 14-6-2022. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/functions/

[12] A. Spina, "Visualizing computational waste in cloud computing," 9 2018, bachelor's Thesis.

[13] P. Vogel, "Computing the cost and waste in cloud computing monitoring," Master's thesis, University of Groningen, 9 2019.

[14] J. Kuhlenkamp and M. Klems, "Costradamus: A cost-tracing system for cloud-based software services," in *ICSOC*, 10 2017, pp. 657–672.

[15] P. Leitner, J. Cito, and E. Stöckli, "Modelling and managing deployment costs of microservice-based cloud applications," in *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, 2016, pp. 165–174.

[16] P. Kumar, P. Singh, S. Chopra, J. S. Sarna, and K. Rawat, "Inspection of cloud computing monitoring tools," in *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, 2017, pp. 361–365.

[17] K. Alhamazani, R. Ranjan, K. Mitra, F. Rabhi, P. P. Jayaraman, S. Khan, A. Guabtni, and V. Bhatnagar, "An overview of the commercial cloud monitoring tools: Research dimensions, design issues, and state-of-the-art," *Computing*, vol. 97, 04 2014.

[18] S. A. De Chaves, R. B. Uriarte, and C. B. Westphall, "Toward an architecture for monitoring private clouds," *IEEE Communications Magazine*, vol. 49, no. 12, pp. 130–137, 2011.

[19] J. Montes, A. Sánchez, B. Memishi, M. S. Pérez, and G. Antoniu, "Gmone: A complete approach to cloud monitoring," *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2026–2040, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X13000496

[20] D. Tovarňák and T. Pitner, "Towards multi-tenant and interoperable monitoring of virtual machines in cloud," in *2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2012, pp. 436–442.

[21] S. Suneja, C. Isci, R. Koller, and E. de Lara, "Touchless and always-on cloud analytics as a service," *IBM Journal of Research and Development*, vol. 60, no. 2-3, pp. 11:1–11:10, 2016.

[22] S. Al-Shammari and A. Al-Yasiri, "Monslar: a middleware for monitoring sla for restful services in cloud computing," in *2015 IEEE 9th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA)*, 2015, pp. 46–50.

[23] J. M. Alcaraz Calero and J. G. Aguado, "Monpaas: An adaptive monitoring platformas a service for cloud computing infrastructures and services," *IEEE Transactions on Services Computing*, vol. 8, no. 1, pp. 65–78, 2015.

[24] M. Lin, Z. Yao, and T. Huang, "A hybrid push protocol for resource monitoring in cloud computing platforms," *Optik*, vol. 127, no. 4, pp. 2007–2011, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030402615017271

[25] Sensu. Accessed: 16-6-2022. [Online]. Available: https://sensu.io/

[26] S. Anithakumari and K. Chandrasekaran, "Monitoring and management of service level agreements in cloud computing," in *2015 International Conference on Cloud and Autonomic Computing*, 2015, pp. 204–207.

[27] M. Al-Ayyoub, M. Daraghmeh, Y. Jararweh, and Q. Althebyan, "Towards improving resource management in cloud systems using a multi-agent framework." *Int. J. Cloud Comput.*, vol. 5, no. 1/2, pp. 112–133, 2016.

[28] AWS Cost Explorer. Accessed: 22-6-2022. [Online]. Available: https://aws.amazon.com/aws-cost-management/aws-cost-explorer/

[29] Microsoft Cost Management. Accessed: 2-6-2022. [Online]. Available: https://azure.microsoft.com/en-us/services/cost-management/

[30] Google Cloud operations suite agents. Accessed: 24-6-2022. [Online]. Available: https://cloud.google.com/monitoring/agent

[31] GCP Cost Management. Accessed: 24-6-2022. [Online]. Available: https://cloud.google.com/cost-management

[32] Datadog. Accessed: 16-6-2022. [Online]. Available: https://www.datadoghq.com/

[33] New Relic. Accessed: 16-6-2022. [Online]. Available: https://newrelic.com/

[34] Sumo Logic. Accessed: 16-6-2022. [Online]. Available: https://www.sumologic.com/

[35] Splunk. Accessed: 16-6-2022. [Online]. Available: https://www.splunk.com/

[36] Nagios. Accessed: 16-6-2022. [Online]. Available: https://www.nagios.org/

[37] Grafana. Accessed: 13-6-2022. [Online]. Available: https://grafana.com/metrics/

[38] Prometheus. Accessed: 13-6-2022. [Online]. Available: https://prometheus.io/docs/introduction/overview/

[39] Grafana and prometheus integration. Accessed: 13-6-2022. [Online]. Available: https://grafana.com/oss/prometheus/

[40] Kibana. Accessed: 14-6-2022. [Online]. Available: https://www.elastic.co/kibana/

[41] Elasticsearch. Accessed: 14-6-2022. [Online]. Available: https://www.elastic.co/elasticsearch/

[42] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, pp. 131–164, 2008.

[43] K. M. Colin Robson, *Real World Research*, 4th ed. Blackwell, 2016.

[44] C. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 557–572, 1999.

[45] AWS Fargate. Accessed: 1-4-2022. [Online]. Available: https://aws.amazon.com/fargate/

[46] GCP Virtual Machine pricing. Accessed: 8-4-2022. [Online]. Available: https://cloud.google.com/compute/vm-instance-pricing

[47] AWS EC2 pricing. Accessed: 8-4-2022. [Online]. Available: https://aws.amazon.com/ec2/pricing/on-demand/

[48] Azure Virtual Machine pricing. Accessed: 8-4-2022. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/virtual-machines/

[49] Cloud Carbon Footprint. Accessed: 2-6-2022. [Online]. Available: https://www.cloudcarbonfootprint.org/

[50] Amazon Kinesis Data Firehose. Accessed: 20-6-2022. [Online]. Available: https://aws.amazon.com/kinesis/data-firehose/

[51] Stream data to an HTTP endpoint with Amazon Kinesis Data Firehose. Accessed: 20-6-2022. [Online]. Available: https://aws.amazon.com/blogs/big-data/stream-data-to-an-http-endpoint-with-amazon-kinesis-data-firehose/

[52] CloudWatch API requests. Accessed: 20-6-2022. [Online]. Available: https://docs.aws.amazon.com/AmazonCloudWatch/latest/APIReference/Welcome.html

[53] AWS Price List API. Accessed: 20-6-2022. [Online]. Available: https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/price-changes.html

[54] Azure Monitor REST API. Accessed: 20-6-2022. [Online]. Available: https://docs.microsoft.com/en-us/rest/api/monitor/

[55] Azure REST API. Accessed: 20-6-2022. [Online]. Available: https://docs.microsoft.com/en-us/rest/api/azure/

[56] Site reliability engineering - introduction. Accessed: 12-5-2022. [Online]. Available: https://sre.google/sre-book/introduction/

A Prepared Interview Questions

1. Could you describe the architecture of your project?
    a. Which Cloud Service Provider (CSP) is being used?
    b. Is it completely cloud based or a hybrid architecture?
    c. Are there any design decisions in the project that cause over-provisioning of cloud resources? (Having a higher level of support or additional features from the CSP)
2. What is the process for making the initial choice in cloud resource selection?
    a. If the focus is not on keeping costs low, is optimization still a big factor in cloud resource selection?
3. What are the reasons to use monitoring tools for this project, if you used any?
    a. Are the reasons for monitoring the same during development and running? If not what are the different focus points?
4. Which monitoring tools are being used in practice in your experience?
    a. What are they (primarily) used for?
    b. Are the same tools used for cost management?
    c. Why are those specific tools used?
    d. Are the techniques used by the tool considered when choosing it?
5. What makes a metric useful or helpful for cost management (in your opinion)?
    a. Which metrics are most important for this project?
    b. Are there any specific requirements that make metrics more important than for other projects?
    c. What visualization techniques do you prefer for these metrics? What makes a metric easy to understand (in a dashboard)?
6. What is lacking in the currently used tools (in your opinion)?
7. To what extend do you think a tool that maps the wasted resources in a cloud application would have added value to the tools that are already being used?
8. How would a cloud resource waste monitoring tool be most useful? Continuously available in a dashboard, a single run script for during development, in the CI/CD pipeline or something else?

B Quotes and Codes

B.1 Interviewee A

ID	Quotes	Codes
A1	”In Azure our API is built on Azure function, so that is serverless, and then we have a Cosmos database in the back end as well and storage and storage tables.”	’Azure Function’, ’Serverless Functions’, ’Database’
A2	”But in the past, I’ve also worked with app services and things like the classic server scenario as well. But your basic web app with a database behind it is something that I’ve worked on quite a bit.”	’Azure’, ’Project Type’
A3	”Most of the things were completely cloud based [and not hybrid-cloud].”	’Public Cloud’
A4	”The feature part [of over-provisioning due to project requirements] is a major consideration, especially if you’re looking at security. So, a lot of the times, especially if you’re doing projects for Deloitte or customers, you might need things like a web application firewall which will be in a higher tier in API management or services like that. So that definitely comes into play that it’s often the case that you need the premium version for certain features, more than you would need it for the performance side of things.”	’Over-Provisioning’, ’Cloud Provider Offerings’
A5	”Usually, we still take a look at the number of customers you expect and what sort of traffic. Make sure that you think that through as well, especially when hitting limits of the storage or other services.”	’Resource Allocation’
A6	”You go through a process where first you set up your test environments which you try and keep as small as possible. But also have the features that you want and then think about how many users do I expect and how much resources would I require for those amounts of requests or users or concurrent users or stuff like that.”	’Resource Allocation’
A7	”That’s [the expected number of users] usually the expectation from the customer side of things on our projects. Like historical usage or just their expectation.”	’Resource Allocation’, ’Project Management’
A8	”Well, I think cost is always a consideration. I’ve never been in the situation where they were like “Oh just make it twice as big for the heck of it”, so to speak.”	’Cost’
A9	”It’s usually mostly features based when you end up with a more expensive solutions in the end. Or the expected growth and needing the higher throughput.”	’Project Constraints’, ’Cost’
A10	”Monitoring tools are always necessary to monitor, in first place the health of your applications, how they’re doing and especially when trying to look at maybe performance issues or other things. Long running requests and stuff like that. When they happen, those monitoring tools play a huge role for getting insight into that.”	’Monitoring Reason’
A11	”Usually, it’s [the reason for using monitoring tools] the error handling or seeing where long running instances or queries, or other things come up, to fix those.”	’Monitoring Reason’
A12	”During development you want as much insight into the errors as possible. During the running you’re more interested in the number of errors and the stability side of things, opposed to diving deep into the details. So, there is a difference, but it’s based on almost the same kind of data and metrics that you would use in both cases. Just a different summary or view in my in my experience.”	’Monitoring Reason’, ’Developer’
A13	”Usually, [I use] the built-in [monitoring] tools in Azure. At least that’s what I’m used to. I haven’t had any experience with any other third-party cost management tools. ... When running projects for clients you usually just look at the monitoring tools available in Azure itself.”	’Provider Native Monitoring Tool’, ’Azure’
A14	”No, that [optimization and cost reduction possibilities] is not actively monitored. It usually comes up when something costs quite a lot.”	’Cost Monitoring’, ’Monitoring Engagement’
A15	”For a project I did for Deloitte internally, the manager would come back saying, “hey, why are we getting a bill for X amount? That seems quite high.” And then we would take a look and we saw that we were running four separate API management instances, which are quite expensive in itself. And we were like “hey, can we fold those for the test environments”, because we had three test environments. Can we fold those into one, so that we’re only using one for all of our test instances instead of three separate ones. So that was really based on like, “hey, why are we getting such a high bill?”. In that case it was really based on the cost that they were actually paying and not because they were looking into what it would cost to run beforehand.”	’Cost’, ’Common Practices’, ’Optimization’
A16	”It [the Azure monitoring tool] breaks down the cost per service and stuff like that. I’ve never had to deal with the actual invoices, so I can’t tell you much about that. It breaks it down really good over time, including which services use quite a lot of resources. I always found it pretty intuitive how they break it down. They have pie charts of all the different things, and you can look at all your different resource groups or subscriptions. And check out what’s burning through your money, so to speak.”	’Provider Native Monitoring Tool’, ’Cost Monitoring’, ’Visualization’
A17	”[A metric is most useful] when it reflects the experience of the end user.”	’Metrics’
A18	”If you’re running an API, or if you’re running a website, the most important thing is that the website is running smoothly for a user or that the API has reasonable response times and stuff like that. Metrics related to that make the most sense, because that’s when you would need to change something. When those metrics change and impact the user or hopefully even before.”	’Metrics’, ’Project Constraints’, ’Project Type’

A19	”I think the ease of tying logging and everything together is still lacking in my opinion. It takes quite a lot of configuration to get things like trace IDs and then you can follow how a request moves through your whole application or your whole architecture. And that can still be quite a hassle to set up. If everything is already in the cloud that is possible, but then getting those relevant graphs out of that combined information. I think that is something that I haven’t seen done well. Something that does that in an easy way for you.”	’Lacking in Monitoring’, ’Graph’
A20	”Seeing all the different parts in your architecture, how they are part of the whole request pipeline. How that all ties together so that you get a view of the overall state of your application. So that you don’t just see that there is a long running request, but that my app is waiting a long time for the database to return the results. You get more fine-grained detail in how your requests are doing instead of just seeing a request is taking 2 seconds to respond. You can already combine all those different services and get more insights into the combined working of all those services in a logical representation. Which is of course really difficult because the architectures are always different, so I see why it isn’t there yet, but that is something that in my view would be lacking.”	’Lacking in Monitoring’
A21	”It’s good to get it [waste monitoring] more on the forefront. Like I said before, right now it’s usually like, “hey, why does it cost so much? Let’s look at the utilization”. It would be good to get that automatically or more regular updates on that.”	’Value of Waste Monitoring’, ’How to Waste Monitor’, ’Monitoring Notification’
A22	”There are of course very simple tools right now. I believe in Azure as well that there is a cost monitoring tool, but I’ve actually never seen a suggestion come from that. That you’ve provisioned a too high app service. I don’t know how those work, but I do know they are there, I’ve never encountered those yet.”	’Cost Monitoring’, ’Azure’, ’Waste Monitoring’
A23	”I think the wasted resource [monitoring tool] doesn’t make sense if you use a single run script unless that script looks at the historical data. If you look at it at deployment, you don’t know how it’s going to run with actual users.”	’How to Waste Monitor’
A24	”So, I think it makes sense to have some sort of continuous monitoring [for wasted resources].”	’How to Waste Monitor’
A25	”Or if you have reports that are generated over a certain time, that would make more sense. Monitoring over time makes sense and if you can tie that into your different deployments and mark those as well in that timeline. That would be a help as well, like if you make a deployment and that made a change to the costs. That would be really interesting to see.”	’How to Waste Monitor’, ’Monitoring Notification’
A26	”On the other hand, if the costs are relatively low then it becomes less interesting of course. So, it generally is more relevant when the costs get a bit higher. But I do think that it would be interesting, and I think it’s most interesting in a changing environment. If you have a stable environment and you’re happy with how it’s performing, I think that there is also a sort of inertia. Or a “why fix it if it ain’t broken?” sort of mentality. Because it costs time of course.”	’Value of Waste Monitoring’, ’How to Waste Monitor’

B.2 Interviewee B

ID	Quotes	Codes
B1	”The architecture consists of basically a monolith application. We don’t have separate deployments over backends, frontends, or micro services.”	’Monolith’, ’Project Type’
B2	”It runs on the Azure cloud, so we make use of native Azure components. Such as ... Application Insights for tracking our bugs and measuring performance.”	’Azure’, ’Provider Native Monitoring Tool’, ’Cloud Provider Offerings’
B3	”It’s completely cloud based, so it runs fully on Azure.”	’Azure’, ’Public Cloud’
B4	”We have different environments like dev, tests, acceptance, staging, and production. So, five in total and especially the lower-level environments aren’t fully utilized, of course. But sometimes for some native Azure resources you need to pick a certain level of capacity and other features, for instance, in order to connect to custom domain or to have certain integration options available. That is the reason why we pick a certain level of provisioning instead of going for lower level. Because for from a feature perspective we need that and not so much from a capacity perspective.”	’Azure’, ’Over-Provisioning’, ’Project Constraints’
B5	”The most important thing is to not have downtime, not serve timeout requests to users. That’s number one, cost is second. We did not really make any active decisions in this regard [keeping cost low]. One other trade off that is sometimes being made is to have your dev, test, or acceptance environment differ from your production environment for cost reasons. If you say we’re going to share resources, we’re just going to have one database instance, and we’re sharing it with like multiple environments in order to save cost. But that’s not what we did. We kept everything as similar as possible except the sizing of the cloud resources. We just use the lowest possible version, so it’s still similar to production.”	’Cost’, ’Common Practices’, ’Optimization’
B6	”So, the monitoring tool that we used was Application Insights. ... [Which is] Azure native. It’s mostly used for tracking bugs and application insights, so really on the application level.”	’Provider Native Monitoring Tool’, ’Azure’, ’Monitoring Reason’
B7	”One thing we also implemented was scaling rules based upon mostly CPU, because that was from our experience always the bottleneck with our instances that we ran.”	’Auto-Scaling’, ’CPU Metric’

B8	”Regarding monitoring tools, we also used the monitor function in Azure itself. To basically define thresholds on latency, on peak usage, on error rates, and we connected that to an external instance of ServiceNow. Which was from the party that did the 24/7 support of the application, so that tickets were automatically created when incidents occurred.”	’Provider Native Monitoring Tool’, ’Azure’, ’Monitoring Reason’
B9	”If we saw too many timeouts or response times that were increasing, we took a look at the resource consumption. Usually what we did was pick a certain size of instance that was, from our experience, sufficient to run the application with a limited number of users. And then have horizontal scaling rules to add more instances. What we saw was when the timeouts occurred and when the latency increased, the CPUs were usually very high in the resources and then we changed our scaling rules.”	’Monitoring Reason’, ’Common Practices’, ’Resource Allocation’, ’Auto-Scaling’, ’CPU Metric’
B10	”We didn’t do continuous performance monitoring on development or test. Only before going live we did an elaborate performance test on all the applications that needed to work together and to serve results to the end user. So, there was no nightly performance test executed on the test environments, so you could preemptively see if performance was degraded by a recent change.”	’Monitoring’, ’Project Management’
B11	”So, what I use Application Insights for is mostly tracking exceptions on an application level. So really on the application layer of the full architecture to see exceptions or warnings. Also, performance of certain pages, so latency of certain URLs that are requested or maybe database queries that don’t perform as they shoot. But not so much looking at the resources themselves, like the performance of the CPU levels, memory consumption of app services because that’s usually what you configure with your scaling rules of your app services. You pick a certain size of app service and that is basically one instance that always runs and that’s what you pay for. So usually, you don’t pick one that’s way too high for your application to run in an idle state because you can use horizontal scaling for those nodes to save cost.”	’Monitoring Engagement’, ’Provider Native Monitoring Tool’, ’Azure’, ’Auto-Scaling’, ’Resource Allocation’, ’Common Practices’
B12	”In Azure you have the Azure costs overview. I don’t know the exact name, but it’s a dashboard in Azure where you can see all the resources and what they cost you. Because cost usually isn’t only driven by capacity or resource allocation.”	’Cost Monitoring’, ’Provider Native Monitoring Tool’, ’Azure’
B13	”I use the cost management overview just to get a feel of what resources incur the highest cost and where we can save some.”	’Cost Monitoring’, ’Monitoring Engagement’
B14	”[I use the Azure native monitoring tools] because it’s out of the box Azure and I’ve never used a different kind of cost overview tool with the Azure cloud. I mostly work with Azure, so I’ve never seen any reason why I should use something else when the native Azure stuff works fine.”	’Cost Monitoring’, ’Provider Native Monitoring Tool’, ’Azure’, ’Reason for Specific Tool’
B15	”One issue that we had for this project was that configuring the correct scaling rules was mostly a trial and error thing. So, we just provisioned some high numbers and then we put in some auto scaling rules, but in the end when peak load occurs, the auto scaling reacted too slowly.”	’Auto-Scaling’, ’Resource Allocation’, ’Over-Provisioning’
B16	”So basically from 4:00 o’clock in the afternoon till midnight. We just increase the nodes, from running one node to 10 nodes. Even though on some afternoons or some evenings it was very quiet, so that was basically not using the capacity in the most efficient way.”	’Project Constraints’, ’Over-Provisioning’, ’Optimization’, ’Auto-Scaling’
B17	”If the auto scaling and provisioning tools on the Azure platform could cope with a sudden influx of users we wouldn’t need to manually scale up at a time and scale down at midnight. Because then you just were able to configure auto scaling rules and then the platform would just figure it out itself, but from our experience that wasn’t sufficient.”	’Auto-Scaling’, ’Over-Provisioning’, ’Optimization’, ’State of the Art’
B18	”What I would like to know is what type of resources are incurring the highest cost? Or what type of environments? Or if you have an app service plan or an app service with that incurs €1000 per month, I want to know what aspect is causing that? Because usually cost is calculated on different levels, like in-going or outgoing traffic or Front Door, maybe it’s the number of routing rules you’ve configured, so it would be nice to see what part of the calculation results in this number of euros on the bill.”	’Cost Monitoring’, ’Lacking in Monitoring’, ’Provider Native Monitoring Tool’, ’Azure’
B19	”Basically, it [the most important metrics] would be the up-time or the number of exceptions in an application because the number one priority is having like a flawless experience for the end user. So maybe the timeouts that that the system has, or the downtime. Even though it’s only one out of 10 nodes that has downtime, that is still like 10% of your users or incoming requests for a certain amount of time. That’s one and then second maybe cost, although that wasn’t the main driver.”	’Metrics’, ’Monitoring Reason’, ’Cost Monitoring’
B20	”We did take a look at it [the cost monitoring data] like once or twice a month, just to peek and scroll through the cost overview to see if nothing weird happened.”	’Cost Monitoring’, ’Monitoring Engagement’
B21	”To go from the base level of users to 10 times as much in just 3 minutes. Yeah, that was something that was hard to configure on the Azure platform, so we took quite some time and looked into all the measurements to optimize that. Like a combination of manual scaling rules and time-based scaling rules and auto-scaling rules.”	’Optimization’, ’Auto-Scaling’, ’Project Constraints’, ’State of the Art’

B22	”It would be nice to have dashboards and maybe you can build this, but we didn’t do it, where you could see all the different nodes that you have. Because we use horizontal scaling, so you might have like 3 nodes in one region and seven in the other, and then to see the health per node. If one node is down you could see this in an individual way, with how much traffic is impacted and if the platform is already dealing with it. So maybe the Azure platform itself should automatically detect if one of our nodes is not performing well and restart it. We don’t need to manually do that, but that would be nice to see that in an overview, and that there are automated actions already being performed.”	’Lacking in Monitoring’, ’Provider Native Monitoring Tool’, ’Azure’
B23	”Seeing the difference between the capacity of the system and the external loads that you place on it. Some way to visualize that would be nice because now on Azure you can see those things a bit, but I don’t think it’s very intuitive.”	’Lacking in Monitoring’, ’Provider Native Monitoring Tool’, ’Azure’, ’Visualization’
B24	”On app service plan level, you can get a graph of CPU consumption, but within one app service you can scale between one node and I say 30 nodes. But sometimes one of those nodes goes corrupt and if you just have an aggregate overview, then you might not notice those things. So more of a visual overview of your entire architecture and the different components and the consumption in each level.”	’CPU Metric’, ’Visualization’, ’Lacking in Monitoring’
B25	”I don’t like working with the Azure portal or like configuring stuff and then having a separate tool running somewhere else that also provides me some information, so ideally everything is integrated with one overview. It should be easy to configure and also with as much automation as possible. So, I don’t need to do much, and it scans the resources and provides some suggestions or some out of the box dashboards that work well with this type of architecture.”	’How to Waste Monitor’, ’Lacking in Monitoring’, ’Developer’
B26	”It would be most useful to run it [a waste monitoring tool] continuously and to have it in some sort of a dashboard. But also, while tracking data, so when at a later moment you get a notification about something that went wrong, you can do some debugging later on and go through the logs and data.”	’How to Waste Monitor’, ’Dashboard’, ’Monitoring Notification’
B27	”[A waste monitoring tool] For integration in the CI/CD pipeline, I don’t think it’s that useful. I think it could only be useful if you run it on a nightly basis. For instance, deploy your application to dedicated resource group, run some tests, and compare the measurements with the night before to see if your performance is still on par. But that’s part of performance testing and to make sure that your application doesn’t use more resources than necessary.”	’Waste Monitoring’, ’CI/CD’
B28	”Well, it is kind of interesting because all cloud providers are kind of slowly moving towards more managed services. First there were virtual machines, then it was PaaS and functions as a service. Recently Azure introduced a container runtime, basically, you run Docker containers in a Kubernetes cluster, but you don’t have to manage the cluster, it’s all been done for you so. The user of the cloud is getting further away from the more low-level stuff.”	’State of the Art’, ’Managed Services’, ’Containers’
B29	”Yeah, you might want to use a certain tool or version of a resource in the cloud because you need a certain feature, not so much because you need the capacity. And then maybe this feature becomes available on all the versions of this resource, so we can scale down to a cheaper tier. But if you don’t follow the release notes all the time you might not notice that. And then you might run for months and months on a size of resource that you don’t need anymore.”	’Over-Provisioning’, ’State of the Art’, ’Developer’, ’Project Management’
B30	”Although it’s not in the interest of the cloud provider to tell you that you are using a more expensive resource than you actually need.”	’Over-Provisioning’, ’State of the Art’, ’Cloud Service Providers’

B.3 Interviewee C

ID	Quotes	Codes
C1	”For the project that I’m currently on, the client runs on AWS.”	’AWS’
C2	”So, at the very core, where most of the workloads run. These are Java based spring boot services. They run on top of Kubernetes, so we use managed EKS, which is the managed container service from AWS.”	’Java’, ’Kubernetes’, ’Managed Services’, ’AWS’, ’Containers’
C3	”But we use Kubernetes and underneath actually Kubernetes just uses EC2 instances which are virtual machines. So, at the very core, most of our Java servers, at the end of the day, are running on regular virtual machines.”	’Kubernetes’, ’EC2’, ’Virtual Machines’, ’Java’
C4	”We try to use managed services as much as possible.”	’Managed Services’, ’Common Practices’, ’State of the Art’
C5	”For example, data is stored in document DB which is a mongoDB compliant database, while the file storage and object storage is done in S3 which is the managed service to store files for from AWS.”	’Database’, ’S3’, ’Managed Services’, ’AWS’, ’Storage’
C6	”For Kafka we are going to use managed Kafka or MSK by AWS.”	’Managed Services’, ’AWS’, ’Kafka’

C7	”In terms of front ends, so the websites, there’s multiple running of them. They actually mostly run in completely managed AWS services. So, we use something called single page applications. So, it’s basically just a set of JavaScript, HTML, CSS. And from the client Web browser, API calls are made, but that is all hosted in S3 with a combination of CloudFront, which is a CDN, content distribution network, to serve those assets to the client.”	’Managed Services’, ’AWS’, ’S3’, ’Storage’
C8	”Everything runs on top of AWS, however the project is within a large financial institution and there are integrations that we have to other systems. Not part of the project, but some of those systems run on premise. So technically, for the project it’s all in the cloud.”	’AWS’, ’Public Cloud’, ’On-Premise’
C9	”Are there design decisions that would directly result in over-provisioning? The answer is no. But in a way, by choosing to run services on Cube, which means at the end of the day those containers and the processes that those containers run are on an actual machine. Which means there’s always some CPU time not being used. So indirectly by choosing to use containers, you already have over capacity in a sense. But it is not the case that a design decision directly resulted in over capacity.”	’Over-Provisioning’, ’Containers’
C10	”That [the process of making the initial choice in cloud resource selection] is twofold, if we take a look at the backend services. It’s Java based, so they’re the type of machines that we run on, and the resource selection is heavily influenced by that. Which means we have machines which are very memory heavy and Java servers are always hungry at boot up. At least Spring boot is, to get CPU and it tends to be very spiky. So, Java is a very large component in choosing that.”	’Resource Allocation’, ’Java’, ’JVM’
C11	”We use managed databases on AWS. But most managed databases, with the exception of DynamoDB, where you actually provision a physical server somewhere, they are completely managed by the cloud provider. But you still have to select a machine with the amount of CPU’s, memory, and storage.”	’Database’, ’Managed Services’, ’DynamoDB’, ’AWS’, ’Resource Allocation’
C12	”We have relatively constant workloads, so it’s [the resource selection] more driven by the number of services we have and database connections, which sort of determines the size of the machine.”	’Resource Allocation’, ’Project Type’
C13	”In our case costs aren’t the biggest driver for making changes. Because the overall costs of running the cloud environment are not seen as a very big component of the total cost.”	’Cost’
C14	”I think overall if you look at the architecture and choices that we made, then what I was primarily looking at is reducing complexity in the environment. That will always take precedence over reducing cost.”	’Optimization’, ’Cost’
C15	”However, recently, we’ve been doing a bunch of upgrades. Which was a good moment to revise a few things and in certain cases, because we have something called a multi account setup in which we run multiple accounts which then run separate environments. And choices have been made to pull things out of those account and move them into a centralized management accounts. So that our multiple environments can use a shared resource.”	’Optimization’
C16	”There are optimizations that you can do to the JVM or even move to a different virtual machine for Java. Those aren’t things that we’re doing. You can very much reduce the memory usage of Java to a tenth of what we have right now, which would reduce costs a lot, but that is not something we currently do.”	’JVM’, ’Optimization’, ’Cost’
C17	”We use a lot of virtual machines or EC2 instances, and the amount that we use is relatively constant because it’s very tightly correlated with the amount of applications that we are running. So, what we do is buy those upfront, so we pay upfront to AWS and that gives you a cost cut.”	’Virtual Machines’, ’EC2’, ’AWS’, ’Cost’
C18	”We run on X86 machines, which are very power-hungry machines if you look at the watt performance. AWS also offers ARM based machines; it’s called Graviton or something. Those machines have a much better ratio between watts and amount of computing power delivered. But we don’t use those either. It’s just a lot of work to reduce complexity and make that shift.”	’Optimization’, ’Environmental Impact’
C19	”In terms of overall monitoring, we use in this case, so this is for everything: that’s infrastructure but also application monitoring. We now heavily rely on a third-party solution called Datadog. Which does everything from pulling in logs, metrics, and other data out of the AWS accounts to create an overview of how the environment is doing performance wise and CPU utilization wise. Just to do basic alerting and detect problems early on, but also have all the data available in one system to troubleshoot and drill down into those problems and find the root cause.”	’Datadog’, ’CPU Metric’, ’Monitoring Reason’
C20	”Outside of that, specifically looking at the costs, we don’t actually use Datadog for that. We actually rely on the AWS tooling. I have a little bit of cost control in place. In terms of checking if the machines that we selected a few months ago are too big for the workloads that we run, it’s not something that we actively monitor.”	’Cost Monitoring’, ’Provider Native Monitoring Tool’, ’AWS’
C21	”The only moment that we really think about whether or not we should address the amount of resources that we run is when we hit performance issues. So, when we need more resources, when we need less is less important. It sometimes does happen, but we don’t actively monitor it.”	’Monitoring Reason’, ’Common Practices’, ’Project Management’
C22	”In terms of what we monitor, whether that is for development or when it’s running live in production is very similar. So, the alerts from the environments are the same, just the process in which they are handled is different. But we do want to make sure that even when a developer develops something on a lower environment and they’re still actively developing it, but they also take into account: “What should we monitor?”, “What should we look out for?”. To make sure that that when it hits production, they don’t all of a sudden find out it doesn’t work. So, we try to actively do that.”	’Monitoring Reason’

C23	”The production is handled by a separate team in this case, so a developer will probably just love a service reporting any errors, whether it’s fast, and then just push the changes into the development branch. Where in the production branch that might be a little bit more expensive and then latency, metrics, and other utilization metrics become more important. Whereas in a lower environment the load is different, so cost is lower.”	’Monitoring Reason’, ’Cost’, ’Metrics’
C24	”In terms of data that flows into Datadog is logs and metrics. If you think of logs, think of the services themselves producing log info, warning, error messages, metrics. It can be very, very broad. Memory usage is a metric, average memory usage is a metric, CPU usage is a metric, the amount of database connections is a metric, so all that data is funneled into Datadog. The purpose of doing that is because you want to look at your error rates in an environment, you want to look at the utilization, whether that is memory or CPU. And there are alerts with thresholds set up to monitor these things, so that if there’s a lot of errors in a certain time frame, alerts will send out. If utilization peaks above certain threshold for amount of time, alerts are sent out. That’s sort of the basic to make sure the customer experience is good.”	’Datadog’, ’Memory Metric’, ’CPU Metric’, ’Monitoring Reason’
C25	”AWS also has monitoring services whether that is CloudWatch, for most of the logs and metrics, up to budgeting and costs. Everything is also on AWS [, as well as Datadog]. AWS actually offers, in terms of capabilities, a full suite.”	’Provider Native Monitoring Tool’, ’AWS’, ’Cost Monitoring’
C26	”We chose Datadog for a number of reasons. When the project started, we had the intention to go live extremely quickly. From nothing being there on day zero, so not even a cloud environment, to the end where we went live in 13 weeks, so a little bit more than three months. If you look at CloudWatch it works fine, but there is quite some effort required in setting up alerts and everything. If you have solutions like Datadog and yes, you pay for it and it’s not as cheap as something like CloudWatch, it’s actually rather expensive. But out of the box it already does it a lot, setting up an alert in Datadog is something any developer can do. So out of the box, the product just does a lot and is very user friendly. Which fit well within our goal of going live very quickly, which means focusing on the actual features for users.”	’Datadog’, ’Reason for Specific Tool’, ’Time Constraints’
C27	”So, for the other things around supporting your whole platform or at least the cloud environment, we just decided to pay extra for a solution [Datadog] which works out of the box really well.”	’Datadog’, ’Cost’
C28	”We are actually switching to CloudWatch now, but that’s one and a half year later.”	’Provider Native Monitoring Tool’, ’AWS’, ’State of the Art’
C29	”Well, Datadog isn’t a tool that is a standard in this organization. So, when we started off, they said “Fine, go ahead and use it”. If this becomes very successful and you guys continue, then we’ll need to revisit the technology stack or the architecture and see what we need to switch out. So, in that sense we’re moving to tools and solutions that in this case this whole organization uses.”	’Datadog’, ’Project Constraints’
C30	”No, [the underlying techniques a monitoring tool uses are] not really [taken into consideration when choosing a tool].”	’Reason for Specific Tool’
C31	”Datadog is a SaaS solution, you don’t run it yourself. I know that they run their workloads on AWS, but that didn’t play a role in the selection of the tool at all. That’s mainly driven by whether we are allowed to use it, where they store their data, if they have certain security requirements, certifications, etc. in place. Are they mature party in the market? Those types of things, but how they run doesn’t matter too much.”	’Datadog’, ’Reason for Specific Tool’, ’Project Constraints’
C32	”Well, I must say we don’t do a lot of monitoring metrics for cost. It’s a little bit difficult to say because if you look at a very basic metric, such as CPU utilization, it depends on where you run your workloads if it correlates with cost.”	’Cost Monitoring’, ’CPU Metric’
C33	”So, when is a metric important? When there is a very close correlation with costs. But it is very difficult because if you run your code as a serverless function, whether that is a Lambda on AWS or an Azure function then you only pay for how long it runs. So, in that case the metric is actual execution time and not the spiky behavior and whether it has over-utilization or not. It’s the run and the execution time of the function.”	’Cost Monitoring’, ’Metrics’, ’Serverless Functions’
C34	”At the end of the day, a lot of stuff that’s monitored is done to make sure that everything was up and running, the availability is high, and its performance is good, so the customers are happy with it. That’s mainly what drives it. But as for if the client asks us to start reporting on the cost drivers, that’s not the case.”	’Monitoring Reason’, ’Cost Monitoring’
C35	”Most metrics are just plotted in a graph over time and value. So generally speaking, looking at such a graphical format is useful. Looking at the raw figures is rarely useful. Just looking at metrics in a graph over time is useful and I must say we only look at the stuff when actual alerts are triggered.”	’Graph’, ’Monitoring Engagement’
C36	”No, [there is] not really [anything lacking in the currently used monitoring tools].”	’Lacking in Monitoring’
C37	”We are switching to CloudWatch, as I said, but that’s not really lacking in terms of capabilities, it is just lacking in how user-friendly it is. Developers don’t like looking at monitoring tooling, they just want to build stuff. Datadog is very good at inviting people to use logging and monitoring tooling, where CloudWatch is a bit more annoying to use.”	’Lacking in Monitoring’, ’Monitoring Engagement’, ’Reason for Specific Tool’, ’Provider Native Monitoring Tool’, ’AWS’, ’Datadog’

C38	”It’s [Datadog] a very big product that integrates with most solutions on the market, or at least the ones that we were running. So, it practically integrates with every AWS service that exists and practically integrates with any third-party service. Whether you run Kafka or RabbitMQ yourself, Datadog will integrate with it, even the cost control side of AWS is in there. But also Azure and Google cloud, it basically, integrates with all of them pretty seamlessly.”	’Datadog’, ’Cloud Provider Offerings’, ’Provider Native Monitoring Tool’, ’Kafka’
C39	”Yes, I think adding that [a waste monitoring tool] dimension of cost to, for example, the CPU utilization can be very useful to see where the biggest cost drivers are and then cut down on them. I can do that [deduce what the waste of cloud resources in the application is] by hand because, roughly speaking, I know what the costs are per month per service. I can correlate that to what I see.”	’Value of Waste Monitoring’, ’How to Waste Monitor’, ’CPU Metric’, ’Cost Monitoring’
C40	”I think Datadog even offers dashboards on it [waste in cloud resources], but I honestly haven’t really looked at them because cost control and utilization aren’t the biggest worrying on the project.”	’Waste Monitoring’, ’Datadog’, ’State of the Art’, ’Project Management’, ’Cost’, ’Optimization’
C41	”I think a tool such as that [a waste monitoring tool] would be most useful if it actively generates insights, based on historical data. Now imagine it sees that a particular server or application has been running, and has never hit full CPU utilization, that would be useful.”	’How to Waste Monitor’, ’CPU Metric’
C42	”Actually, this [notify you about over-provisioned resources] is something that AWS will actually do, they will notify you that certain machines are over-provisioned and if you look at the management console, they will actually actively inform you of these things. So, I think the most important part is actively informing you of actions to take to reduce the waste.”	’How to Waste Monitor’, ’Provider Native Monitoring Tool’, ’AWS’, ’State of the Art’, ’Over-Provisioning’, ’Optimization’, ’Monitoring Notification’
C43	”So, it [how to monitor waste] depends a bit on the situation, for example what I said at the beginning. We can also move our Java services to ARM based processors and rip out the JVM and replace it with something else. Those are of course very detailed insights that are very difficult to automatically generate.”	’How to Waste Monitor’, ’Optimization’, ’JVM’

B.4 Interviewee D

ID	Quotes	Codes
D1	”AWS [is the cloud service provider being used]”	’AWS’
D2	”It’s almost completely cloud based, cloud native, serverless. So, the only thing we provision are databases.”	’Public Cloud’, ’Serverless’, ’Database’
D3	”It’s [serverless is] everything from the serverless stack of AWS. It’s Lambda, so serverless functions, containers in Fargate, which is also being marked as serverless by AWS.”	’Lambda’, ’Serverless Functions’, ’Containers’, ’Managed Services’, ’State of the Art’
D4	”And for instance, Dynamo is the for data storage and S3.”	’DynamoDB’, ’S3’, ’Database’, ’Storage’
D5	”In production, what we do for every container we spin up, we always spin up two. Even when it’s not necessary, but that’s for high availability.”	’Containers’, ’Over-Provisioning’, ’Project Constraints’, ’Resource Allocation’
D6	”We also always provision a read replica for every database, also for our high availability.”	’Over-Provisioning’, ’Project Constraints’, ’Resource Allocation’, ’Database’
D7	”I think that’s about it, because the rest of the services we use are always scaling up and scaling down whenever they need to.”	’Auto-Scaling’
D8	”We’ve got some guidelines [for making the initial choice of cloud resources] provided by the company that hired us for that. They believe in AWS as their preferred partner, and they want us to leverage as much native services of AWS as we can. Being as high in the AWS service stack as possible.”	’AWS’, ’Project Constraints’
D9	”For instance, if we need compute, put it in a Lambda function. If it doesn’t fit for whatever reason, then put it in container. If that fails, then we should go back to the drawing board and if that fails, we could always provision an EC2 instance and do stuff on that. But for this project we didn’t have to resort to that yet.”	’Lambda’, ’Serverless Functions’, ’Containers’, ’EC2’, ’Virtual Machines’, ’State of the Art’
D10	”For some services in our project, the cost optimization is definitely something we always have in mind.”	’Cost Monitoring’, ’Optimization’, ’Cost’

D11	”For instance, putting a webservice in a Lambda function or in a container. It can be a cost-based decision to not put it in a Lambda function, but in a container. Because a container can be cheaper in the end, depending on the number of invocations you need. So, we do take that into account.”	’Lambda’, ’Serverless Functions’, ’Containers’, ’Cost’
D12	”On the other hand, we put everything in Dynamo DB tables and we always leave them in on-demand mode, which is really very costly over time.”	’DynamoDB’, ’Cost’, ’Database’
D13	”Everything is being monitored via Datadog, which is a third-party product, so it’s not AWS native. That has to do with the contract between the company I work for and the company that hired us. They demand that everything is being in monitored via Datadog, because that hooks up directly into our alerting systems.”	’Datadog’, ’3rd Party Monitoring Tool’, ’Project Constraints’, ’Reason for Specific Tool’
D14	”The monitoring is mainly done for performance and not for cost. ... [So] error rates, latency, that kind of stuff.”	’Monitoring Reason’, ’Cost Monitoring’
D15	”Well, I’m not really a developer, but I know the developers are also using the monitoring tools while developing, to see the failure rates on their external calls for instance. But most of the times the monitoring is being used in the running phase for troubleshooting and that kind of stuff.”	’Monitoring Reason’
D16	”Of course, CloudWatch is being used, but all the CloudWatch metrics are being forwarded to Datadog and most of the CloudWatch logging, at least the error and the warning logging is being forwarded to Splunk for analysis.”	’Provider Native Monitoring Tool’, ’AWS’, ’Datadog’, ’Splunk’
D17	”But if you need to deep dive, we don’t have everything in Splunk because that’s way too costly, then you have to go back to the CloudWatch log group and have a look over there. But it’s always harder to find a certain detail in CloudWatch than in Splunk.”	’Splunk’, ’Provider Native Monitoring Tool’, ’AWS’, ’Reason for Specific Tool’, ’Cost’
D18	”[The primary use of the monitoring tools is] operational monitoring, so are we still in the green? So, is my service or my function behaving correctly?”	’Monitoring Reason’
D19	”For cost management we only look at the basics. We just take a look at the bill and do some anomaly detection by hand for that. Because a lot of things we use in AWS are pay-as-you-go, so if you have a lot of messages being processed by the application the bill will scale up linearly with it. ... You can say the bill is high, but that’s just a feeling and it is what it is. But as long as we can always explain it, it’s fine.”	’Cost Monitoring’, ’Cost’
D20	”Because that’s [the reason to use Datadog as monitoring tool] something which came out of the contract between the two companies, so it was just a given.”	’Reason for Specific Tool’, ’Datadog’, ’Project Constraints’
D21	”What we use metrics for in regard to cost management is: if we see huge spikes of some AWS services in the expense reports. Then we use Datadog to see which service had spiked and try to find out why it spiked.”	’Metrics’, ’Cost Monitoring’, ’Datadog’, ’Provider Native Monitoring Tool’, ’AWS’
D22	”Because we are currently in a migration project it’s [cost management and optimization] not our main focus. I’m pretty sure that whenever we’re ready with migrating the project from the old provider to AWS this will be a bigger thing. Because currently every developer in our team is working his butt off to make the migration possible within the least amount of time. But this will definitely be a thing in the future.”	’Project Type’, ’Cost Monitoring’, ’Optimization’, ’Time Constraints’
D23	”In the real time project, we spend a fraction of the cost of what the data warehousing part of the project costs. So, we we’re not a cost driver of the whole project.”	’Cost’
D24	”We use the Datadog dashboarding part for that [visualizing metrics]. ... [It has] graphs and you can also plot metric values and that kind of stuff, so we’ve got some overviews.”	’Dashboard’, ’Graph’, ’Metrics’
D25	”No, [there is] not really [anything lacking in the currently used monitoring tools], but I know there are some functional requirements which we can’t deliver currently. But that’s also because we don’t have all the metrics delivered to us, via the software, in the correct way. It’s just not focused at the moment because if we have to fix that, then we have to assign tasks to developers who are already way too busy.”	’Lacking in Monitoring’, ’Metrics’, ’Time Constraints’, ’Project Constraints’
D26	”I think it would be nice to have [a tool that maps the waste of resources in cloud applications], because, if you have a tool that is able to guide you to some waste in a clear way then it would be easy to determine what you should do or where you should focus on for cost management.”	’Value of Waste Monitoring’, ’Cost Monitoring’
D27	”I think continuously running would be most useful [for a cloud resource waste monitoring tool] because what we see a lot is that, for instance, Lambda functions or containers being developed worked perfectly during the development phase and in the integration testing and in the pipelines. But as soon as you put a real load on it, then you see some parameters have to be changed. For instance, more compute or more memory, that kind of stuff.”	’How to Waste Monitor’, ’Lambda’, ’Serverless Functions’, ’Containers’
D28	”When we started, we were very keen on the cost management stuff, for instance, at the time if you ran a Lambda function, you always had to pay 100 milliseconds of compute time for its execution. Currently that’s not the case anymore. So, what we did in our whole processing environment is create Lambda functions and group them in such a way that we can predict that they are as close as possible to the 100 milliseconds runtime of Lambda for that batch. To process as much as possible for the same amount of money. But that kind of stuff is not viable anymore because AWS is currently changing the way they are billing stuff, so that’s also something to keep in mind.”	’Cost Monitoring’, ’Lambda’, ’Serverless Functions’, ’Resource Allocation’, ’Cost’

D29	”Also, in the tool that you are going to build. It has to be flexible, so you can adjust it according to AWS billing specifications. Because AWS is always looking at the market, “What’s Google doing? What’s Azure doing?”. They always want to stay competitive, so the way they charge you changes well every now and then.”	’How to Waste Monitor’, ’State of the Art’, ’AWS’
D30	”Sometimes during development when you’re focusing on optimizing one point of the solution, they [AWS] changed their way of billing and it’s not viable anymore. So, at that point you just leave it be, because optimizing will not bring you that much value anymore.”	’Optimization’, ’State of the Art’
D31	”I think you should always do that [try to optimize for cost efficiency] because you have to be on the ball, cost wise, because if you don’t then you are screwed after a certain amount of time. If you don’t keep up with AWS, the costs will start piling up and you don’t want that.”	’Optimization’, ’State of the Art’, ’Common Practices’, ’Cost Monitoring’
D32	”I check the bill once a month to see if there is anything weird going on. But I can’t see everything seeing as the company that hired us has pre-paid certain services, so it’s not a complete overview. Staying within sensible ranges is much more important.”	’Cost Monitoring’, ’Monitoring Engagement’, ’Project Constraints’, ’Cost’
D33	”Some services do have over-provisioning due to the cost of IO operations. If we run a batch job a couple of times a day, where the database has to do a full scan of the storage, it becomes very costly due to the many IO calls needed. When we doubled the size of the memory for the database the full scan was no longer needed. So even though the cost for memory was higher, the bottom line was lower since we didn’t need to make so many IO calls anymore.”	’Over-Provisioning’, ’Database’, ’Cost Monitoring’, ’IOPS Metric’, ’Memory Metric’, ’Optimization’

B.5 Interviewee E

ID	Quotes	Codes
E1	”Most of them [the projects] have a focus on building small reusable components, which is often called micro-services.”	’Microservices’
E2	”We usually use event driven as well [as microservices] because the world is moving more and more away from the request/reply kind of mechanics because of the scalability and flexibility limitations it has.”	’Architecture’, ’State of the Art’
E3	”We’ve done projects where we run everything in Docker containers with it being Java microservices.”	’Docker’, ’Microservices’, ’Java’, ’JVM’
E4	”We’ve done serverless data processing architectures like with glue jobs on AWS, which is basically a managed version of Spark. And we also do a lot of serverless stuff, like Lambda, SQS, SNS. Which is sort of the serverless/event-driven stack on AWS, so that’s how I would generally describe the architecture.”	’Managed Services’, ’Serverless Functions’, ’Lambda’, ’SQS/SNS’, ’Queue’
E5	”At the moment I’m mostly doing AWS projects, but we have colleagues focusing on all three major cloud providers. Being AWS, Azure, and GCP. The majority of our projects are currently either Azure or AWS, GCP is more of an upcoming provider. I think that’s also applicable in general to the market in the Netherlands. But my focus personally is primarily AWS.”	’AWS’, ’Azure’, ’GCP’, ’State of the Art’
E6	”Most of the Greenfield stuff that we build, which is the majority of what we do, uses the public-cloud only. But there are, of course, also a lot of clients that have stuff running on-premise, in data centers, that they want to bring to the cloud. And what you usually see then is that you, at least temporarily, have a hybrid architecture as you’re moving component by component to the cloud. So, I would say the majority is public cloud only, but we do encounter hybrid setups.”	’Project Type’, ’Public Cloud’, ’On-Premise’, ’Hybrid Cloud’, ’Project Constraints’
E7	”Especially if you look at the running containers, it’s very easy to over-provision those clusters.”	’Containers’, ’Over-Provisioning’
E8	”But also, not investing in stuff like elastic scaling. Which means that by default you sort of over-provision your microservices to take more memory and CPU, so that you know you don’t immediately hit bottlenecks.”	’Auto-Scaling’, ’Over-Provisioning’, ’Microservices’
E9	”In the more serverless environments there is less over-provisioning on our implementation side because most of the sizing and provisioning of cloud infrastructure is done by the cloud providers themselves”	’Over-Provisioning’, ’Managed Services’, ’Cloud Service Providers’
E10	”We’ve had some cases where things just had to get done and we didn’t have time to fix all the performance optimizations for the first version, but we tackled it in the second version.”	’Optimization’, ’Time Constraints’, ’Project Constraints’
E11	”But generally, I would say most of the over-provisioning challenges are in the container space.”	’Over-Provisioning’, ’Containers’
E12	”Of course, you look at the demand and what you expect in terms of concurrent users. Do you expect any big fluctuations? How predictable is the traffic?”	’Resource Allocation’, ’Common Practices’

E13	”We look at efficiency of the software that we’re running. For example, we built a lot of stuff on Java and Java is not known to be the most resource efficient environment due to the JVM being kind of a memory hog. Of course, there are optimized versions of it, but you don’t always spend the necessary time to optimize to that extent. To shave off a couple megabytes of memory usage per microservice doesn’t really make sense until you run thousands of them. So, we generally look at expected traffic and how things run locally.”	’JVM’, ’Time Constraints’, ’Optimization’, ’Over-Provisioning’
E14	”You can give it a default amount of CPU and memory, and there is also a burst amount that you can set. ... So that’s usually how we how we do resource allocation in container environments.”	’Resource Allocation’, ’Common Practices’, ’Containers’
E15	”Our mindset has shifted a little bit from just using containers because every developer understands them and they are pretty easy to work with, to using serverless by default and only using containers when necessary.”	’Containers’, ’State of the Art’, ’Serverless’
E16	”A lot of our more recent projects are using more and more serverless compared to older projects, because the costs are lower, and the efficiency is a lot higher. I think that’s also because the serverless landscape has increased quite a lot in its usefulness.”	’Serverless’, ’State of the Art’, ’Cloud Provider Offerings’, ’Cost’
E17	”Both of them [serverless functions and managed services]. For example, if you look at purely serverless functions, it’s mainly Lambda functions and maybe Glue jobs, but we increasingly use SNS and SQS, which are like the message brokers compared to something like Apache Kafka. Which you would run on your own clusters and which you need to provision yourself. So again, you usually run into a little bit of over-provisioning because you don’t want to babysit the clusters 24/7. So, I think in general we go more towards the serverless now.”	’Managed Services’, ’Serverless Functions’, ’Lambda’, ’Queue’, ’SQS/SNS’, ’State of the Art’
E18	”Of course, observability is really important, especially in a distributed architecture. Also for performance, making sure that SLAs are kept. For security, to make sure that the zero trust principles are adhered to, so that every component looks at the request before processing it and make sure that it’s valid. But also, to make sure that that we don’t over- or under- provision to extreme degrees.”	’Monitoring Reason’, ’SLA’, ’Over-Provisioning’
E19	”For the development, the focus is more on the functional side of things. Checking whether it works, if a request is behaving like we expect it to, are there any security alerts going off, that kind of stuff. I think the efficiency aspect kicks in at the operational level when you need to run it in production.”	’Monitoring Reason’
E20	”It’s a mix between cloud monitoring solutions from the cloud vendors and 3rd party tools. We used a lot of third-party tools in earlier projects.”	’Provider Native Monitoring Tool’, ’3rd Party Monitoring Tool’, ’State of the Art’
E21	”For example, we send all the metric loggings from containers using tools like Fluentd. Where microservices would just log to the standard out, Fluentd would gather it all up and usually push it to a tool like Datadog. Datadog can then display all the metrics and all the logs in neat little dashboards and overviews for the developers to look at and for the operations teams to have a look at.”	’Dashboard’, ’3rd Party Monitoring Tool’, ’Datadog’, ’Metrics’, ’Reason for Specific Tool’
E22	”For the security side of things, they somehow really prefer Splunk. For some of our bigger projects we built a dedicated security monitoring stack and that was based on Splunk instead of Datadog.”	’Reason for Specific Tool’, ’Monitoring Reason’, ’Splunk’
E23	”But in more recent projects we use the cloud native offerings like CloudWatch more and more. With CloudWatch it’s just out of the box, and especially if you move more towards serverless it integrates really well and is relatively easy to use. Feature wise they also made quite a big step.”	’Provider Native Monitoring Tool’, ’AWS’, ’State of the Art’
E24	”And if you look at more complicated use-cases like, distributed-tracing it’s all implemented rather well now by the cloud vendors themselves. Tools like New Relic and Datadog, that used to be far ahead of what the cloud vendors did themselves, have been caught up to in a large degree. So, it makes more sense now to use the cloud vendors their own tooling.”	’Provider Native Monitoring Tool’, ’3rd Party Monitoring Tool’, ’State of the Art’, ’Reason for Specific Tool’
E25	”I think we rely a bit more on what the cloud vendors offers for this [cost management monitoring]. For example, AWS has a lot of things like billing dashboards, cost alerts, and that kind of stuff.”	’Cost Monitoring’, ’Provider Native Monitoring Tool’, ’AWS’
E26	”Well, obviously the metrics on CPU, memory, and disk are primary to cost allocation.”	’CPU Metric’, ’Memory Metric’, ’Storage Metric’, ’Cost Monitoring’
E27	”I think performance metrics could also be really useful. If there is a very predictable traffic pattern, usually you can do some elastic scaling around that, to make sure that in peak moments you have a bit more provisioned than normal. Because if you have very erratic traffic, you run into some additional challenges where you need to scale up and down very fast.”	’Auto-Scaling’, ’Metrics’
E28	”Usually start off with a graph because it’s just easier to see patterns on a graph in general. But as soon as you want to look at why something is happening, you usually drill down into the graph and look at individual log messages or some deeper indicator.”	’Graph’, ’Common Practices’
E29	”So, on this sort of journey from observation to solved, you use different visualization techniques and you usually start off with a more aggregated graph representation. But as you want to know more, you drill down more into table and list-based representations of stuff.”	’Visualization’, ’Common Practices’

E30	"In my experience, it's still quite difficult to bring the whole picture together. We usually have logs, metrics, and distributed tracing. But an all-in-one holistic interface that can do it all is still quite rare. Usually, you have metrics in one and logs in another and you then use the correlation IDs. Then you go to another tool to look at the breakdown into a tracing kind of thing, where you can see all the individual hops."	'Lacking in Monitoring'
E31	"So, the tools themselves are fine, but I think from an efficiency perspective of someone looking at a problem, there's still a lot to gain. Having just one tool that does it all still feels rare, depending on the stack you're using, I would say."	'Lacking in Monitoring'
E32	"I don't think the developer necessarily cares as much in general about the under-utilization or over-provisioning."	'Value of Waste Monitoring', 'Developer', 'Monitoring Engagement', 'Over-Provisioning'
E33	"But I think from a client perspective, they're paying for stuff that they're not using. I think Deloitte, as a whole, also feels responsible for doing the right thing in society and over-provisioning clouds is really bad for the environment. So, I would say there is definitely value there from a societal and cost perspective viewpoint."	'Project Management', 'Cost', 'Environmental Impact', 'Over-Provisioning', 'Value of Waste Monitoring'
E34	"I would generally say [a cloud resource waste monitoring tool would be most useful] as a dashboard."	'How to Waste Monitor', 'Dashboard'
E35	"I could also imagine that at some point you could integrate something in CI/CD. ... You could also do a scan on the sort of codebase and look at your Kubernetes manifest and give of an alert if it is quite a lot of resources for a typical microservice built in this tech stack. And then link to some possibilities to optimize, so give some proactive feedback."	'How to Waste Monitor', 'Optimization', 'CI/CD'
E36	"I also see an added benefit in the developer side of things, but I think a dashboard is where it all starts."	'How to Waste Monitor', 'Dashboard'
E37	"Seeing as the mindset is sort of shifting from doing everything in containers, to thinking about whether we can do it using serverless. Designing the system from the ground up to be more efficient in utilizing resources is of course better than solving it after the fact. And it is also cheaper."	'State of the Art', 'Containers', 'Serverless', 'Resource Allocation'
E38	"So, I would say having some way to integrate [a waste monitoring tool] into the way that we design [the architecture of projects] would be really, really helpful as well."	'How to Waste Monitor', 'Architecture'
E39	"It [how often an engineer looks at monitoring tools] depends on the engineer how much they care about the entire picture."	'Monitoring Engagement', 'Developer', 'Monitoring'

B.6 Interviewee F

ID	Quotes	Codes
F1	"I have two projects I could talk about so I can do both. Actually, they're probably interesting to contrast because one is fully serverless and the other one is not fully serverless."	'Architecture', 'Serverless', 'Virtual Machines'
F2	"I'll start with the one that is not serverless, which was built on a platform called the Open Data Platform, which is something that we have worked on within Deloitte. It is a platform that has at its centerpiece, Kubernetes and containers. And the idea of the platform was to run on AWS, but also, Azure, GCP, or on-premise. So, using containers was a very good technology for us to use, because it meant that we could build something that is portable."	'Containers', 'Kubernetes', 'AWS', 'Azure', 'GCP', 'On-Premise', 'Public Cloud'
F3	"We really centered on Kubernetes to run these big sort of Kubernetes clusters. I think in our environment we had at least one Kubernetes cluster for our CI/CD and then for our application clusters, for each environment, we had two. ... And then we would have a whole bunch of EC2 instances where the microservices are running on."	'Virtual Machines', 'EC2', 'Kubernetes', 'CI/CD', 'Microservices'
F4	"We used AWS managed Kafka, which is serverless, but we can talk about the waste associated with that in a minute."	'Managed Services', 'Kafka', 'AWS'
F5	"And several databases, which were massively over-provisioned, for different reasons. We can talk about that in a second, but a lot of databases. RDS databases, Redshift databases, and some Neptune databases, which are graph databases."	'Over-Provisioning', 'Database', 'Infrastructure'
F6	"For the other client that I'm working with, we have adopted this same platform, but we have tried to be completely serverless. So, in this new product we're building, we currently have zero EC2 instances. So, we're only using AWS Lambda, SQS and SNS for queuing, and we're using DynamoDB for our storage of data. Those are the main services that we're using in this architecture, so it's fully serverless."	'Serverless Functions', 'Lambda', 'SQS/SNS', 'DynamoDB', 'Database', 'Queue'

F7	”Well for the first project that I described, we are using some third-party services for logging, monitoring, and for CI/CD. But the bulk of the cost and the platform is all on AWS.”	’3rd Party Monitoring Tool’, ’CI/CD’, ’Cost’, ’AWS’, ’Public Cloud’
F8	”For the first project that I mentioned, there are definitely design decisions that affect over-provisioning! In the project we were receiving large amounts of data. And we were doing a lot of processing on that data as it arrived, but it would only arrive every three months. So, when get mass amounts of data, all of our compute resources are being used very heavily for a period of a week or maybe two. After which the processing would really slow down. Leaving us with a lot of idle infrastructure that’s just sitting there waiting for the next batch to come in. There definitely would have been ways for us to scale that down, but due to different things people were doing with testing, it sort of made sense to keep some of them running. But there was a lot of over-provisioning there.”	’Over-Provisioning’, ’Optimization’, ’Project Constraints’, ’Project Type’
F9	”The other issue from a design decision was that we were in a little bit of unknown territory with the amount of data that we were receiving. So, whenever we ran into performance issues, the answer was always: “Just give it more CPU. Give it more RAM.” And we would say: “Are you sure? It will cost this amount” and the response was always “Don’t worry about the cost.” So, cost just wasn’t that much of a concern. We could just scale things up massively, but once you scale things up and it works, there is no incentive to scale it back down. I think that’s one of the big things that we learned.”	’Project Type’, ’Project Constraints’, ’Over-Provisioning’, ’Common Practices’, ’Cost’
F10	”We would have these massive databases with large throughput resources provisioned. That was one of the big things we were paying for, the provisioned throughput. There was a lot of money being spent on that. But we weren’t always monitoring it to scale it down after we massively scaled up.”	’Database’, ’Cost’, ’Cost Monitoring’, ’Provisioning’, ’Optimization’
F11	”The other factor involved was, that we would do this on production and then they said: “We want to do the same things on our dev environment.” “Can you make dev the same as production?” Resulting in these massively over provisioned databases, times four, because you have them for each of your environments. ... I think they’ve all been scaled down now, but there was a period of time where we had over-provisioning in one environment and that led to over-provisioning in all the other environments as well.”	’Project Management’, ’Project Constraints’, ’Over-Provisioning’
F12	”They’ve actually been on a journey of making it [the project] serverless, and they’ve been working on that for around four months now. I think there’s been a lot of really great progress there to prevent this thing from happening. But it’s very easy to get yourself in a situation where you have over-provisioning on multiple levels of your application.”	’Over-Provisioning’, ’Serverless’
F13	”I would like to say that there’s a wonderful process of gathering performance metrics, but when you’re first starting out, usually you just give it the resources you think will work and see how that goes.”	’Metrics’, ’Resource Allocation’, ’Common Practices’
F14	”If I’m provisioning an instance on AWS and I don’t really know what I’ll need, I’ll provision an “m5.large” or an “r5.large”. If I don’t know if my application is going to be particularly resource intensive or compute intensive. If I do know the answer to that, then it gives some direction for deciding on what type of instance to use. But usually it’s only at runtime, by closely following your metrics, that you work out what the best instance is.”	’Metrics’, ’Resource Allocation’, ’Common Practices’
F15	”So, I think our initial approach for things that we’ve built, has been to stick with what we know, which is “m5.large” instances. Or “r5.large” instances because we run a lot of Java applications, and Java is, from a memory point of view, not always the most optimized. So, we found that we sometimes get better performance from “r” instances.”	’Resource Allocation’, ’Common Practices’, ’Optimization’, ’Java’, ’JVM’
F16	”But ideally you would run a test of every single different instance and instance type and class and see which is the most performant. But we’ve never really had the time to do that, and there’s never been much encouragement from management.”	’Project Management’, ’Resource Allocation’, ’Time Constraints’
F17	”There’s not always an incentive [to optimize]. If no one complains about the cost of things, and no one wants it to be the most performant it can be, and if the answer is ‘just throw money at it”, then you never you never have the time to go through the proper steps. And that leads to waste.”	’Optimization’, ’Project Management’, ’Cost’
F18	”I think once you’re told that cost doesn’t matter... And that’s one of the problems with cloud. People always sell cloud as if you can scale it by just turning a dial up to 11 and then it’ll perform the way you want it to. And that’s not always the case, so I think that that’s a big factor.”	’Cost’, ’Project Management’
F19	”We were running into some big performance issues with one of our databases and the answer then was just throw more CPU and RAM at it, just give it more money basically. We did that and it didn’t fix our problems. What we found was that there needed to be more tuning of the database to get the performance out of it that that we needed, and the solution was not to throw money at it.”	’Project Management’, ’Cost’, ’Developer’, ’Database’
F20	”The solution was to throw less money at it and to modify some of the configuration of it. Which was not something that AWS showed us in the console, but we had to go into the details ourselves. We had to have some knowledge of how the database worked and do the sort of work that a very traditional database administrator would do. Which again is one of the selling points of cloud, they say you don’t need to worry about those things because you just press a button, and it scales. And for us that wasn’t necessarily the case.”	’Lacking in Monitoring’, ’Database’
F21	”So, even when you have that “thrown money at it” approach, sometimes you do actually need to dive into the details on performance and monitoring.”	’Cost’, ’Monitoring Reason’
F22	”[The main reason for monitoring is] Performance bottlenecks, knowing how to size your instances.”	’Monitoring Reason’

F23	”If things start dying and your application starts falling apart, then you want to know why. If you don’t have metrics, you have no way of judging that.”	’Metrics’
F24	”Things will break at some point, hardware will always fail, so you’ll never have 100% up-time. The question is: when you don’t have 100% up-time, you want to know why, and you need metrics for that.”	’Monitoring Reason’, ’Metrics’
F25	”I guess it [monitoring] matters more in the operational phase, because then you’re dealing with real money, real customers using a product. So, if it goes down then there might be money, reputation, goodwill, or any of those other sorts of metrics that can be can really hurt an organization or a product.”	’Monitoring Reason’
F26	”In development it [monitoring] can matter on several levels. One, because you want your developers to be happy and you want their environments to be working correctly. But also, you want to be able to know how your applications will do in production.”	’Monitoring Reason’
F27	”So, if you’re launching some product and you’re testing it on dev and it seems to work, but then you realize if a million people try and use this at the same time it might break. Sometimes you only know that when you launch it, so you want to simulate that as best you can in dev. So, I think it definitely matters in dev, but if you get it wrong, it’s not going to destroy your company, whereas if you get it wrong in production then it really can.”	’Monitoring Reason’
F28	”Traditionally I’ve used some external tools, Datadog we’ve used quite a bit for monitoring. Which is because the monitoring on AWS, particularly in the container and Kubernetes space, has not always been great.”	’Datadog’, ’Containers’, ’Kubernetes’, ’AWS’, ’Provider Native Monitoring Tool’, ’Reason for Specific Tool’, ’Lacking in Monitoring’
F29	”But I think they’ve [AWS] really upped their game in the last couple of years or so. So increasingly we’re using more AWS CloudWatch and Container Insights for monitoring.”	’Provider Native Monitoring Tool’, ’AWS’, ’Reason for Specific Tool’
F30	”AWS has performance monitoring and enhanced monitoring for RDS for databases and the level of detail that you get is very high. You get very useful metrics from that.”	’Provider Native Monitoring Tool’, ’AWS’, ’Database’, ’Metrics’
F31	”Sometimes interpreting those metrics is very difficult. You need to have specialized knowledge to be able to understand what some of those metrics are telling you, which sometimes can be a difficulty in itself, just understanding the data that is being thrown at you.”	’Metrics’, ’Developer’
F32	”I haven’t used it, but AWS recently launched a managed Grafana service, Grafana and Prometheus I think are also very standard monitoring tools that we use, and I think a section of the general market as well.”	’Provider Native Monitoring Tool’, ’3rd Party Monitoring Tool’, ’Managed Services’, ’AWS’
F33	”I think in the container world, the sort of metrics that we’re generally interested in, are CPU, memory, and networking.”	’Containers’, ’CPU Metric’, ’Memory Metric’, ’Networking Metric’
F34	”I think there’s another layer of metrics which is also about security monitoring, and sometimes that border between security monitoring and performance monitoring can be a very thin line. Because if your application starts behaving very erratically it could be because maybe it’s not performing well because it’s been compromised. Security people can be very interested in those sorts of metrics as well.”	’Monitoring Reason’
F35	”So, I think CPU, memory, network are some of the key things. I think for databases those metrics are also very useful, but read and write operations, IOPS, are very key for us, particularly with that example I mentioned before.”	’CPU Metric’, ’Memory Metric’, ’Networking Metric’, ’IOPS Metric’, ’Database’
F36	”I think AWS provides a lot of these things [cost management tools] out of the box for you.”	’Cost Monitoring’, ’Provider Native Monitoring Tool’, ’AWS’
F37	”I think for a lot of stationary infrastructure, like once you have provisioned your EC2 instances and RDS databases the cost is pretty static, so you already know what your cost is going to be. But then it’s only once you interpret those graphs and then resize your instances that you’ll actually see a difference in the cost there.”	’EC2’, ’Virtual Machines’, ’Database’, ’Cost’, ’Cost Monitoring’
F38	”At least with AWS, it’s very easy for you to be able to look at a bill and it gives you an itemized list of exactly what the charges were for, whether it was for databases, EC2 instances, or VMs, and then you can sort of dive into that more.”	’Cost Monitoring’, ’AWS’, ’Provider Native Monitoring Tool’

F39	”I think the stack that I always preferred was Datadog for the collection of metrics and Sumologic. We use those external tools in part because the platform that we were building had to be cloud agnostic. So, we actually didn’t want to use AWS specific tools for monitoring, so that was a very conscious decision.”	’3rd Party Monitoring Tool’, ’Datadog’, ’Provider Native Monitoring Tool’, ’Project Constraints’, ’Reason for Specific Tool’
F40	”But we also found that particularly in the space of containers, the level of monitoring that we got, in AWS at least, just wasn’t that great. Like the insights into the performance of containers, if we want to know CPU and memory usage of containers, or even just to view the logs of containers. All of that was very difficult to set up out of the box and it was very expensive to set up as well.”	’Containers’, ’Provider Native Monitoring Tool’, ’AWS’, ’CPU Metric’, ’Memory Metric’, ’Cost’, ’Reason for Specific Tool’, ’Lacking in Monitoring’
F41	”It was definitely cheaper to use something like Datadog and Sumologic [than AWS native monitoring tools].”	’3rd Party Monitoring Tool’, ’Datadog’, ’Reason for Specific Tool’
F42	”So, it [the reason to use third-party monitoring tools] was a combination of it being cheaper and cloud agnostic.”	’3rd Party Monitoring Tool’, ’Cost’, ’Reason for Specific Tool’
F43	”There was also an argument made at the time, if AWS goes down and you lose your application, you also lose access to your log and your metrics, so having them at a third-party means that you have another copy of it in the event where things go down. So, there is an availability argument to be made as well.”	’3rd Party Monitoring Tool’, ’Provider Native Monitoring Tool’, ’Reason for Specific Tool’
F44	”We wanted to build something that could be used anywhere. Organizations will have preferences on which cloud provider they want to use, or maybe they don’t want to use public cloud and use on-premise. But we often speak to organizations that don’t want to use AWS or don’t want to use Microsoft, for various different reasons, but it can be very deep-seeded reasons why organizations won’t pick particular cloud providers.”	’Cloud Service Providers’, ’Public Cloud’, ’On-Premise’
F45	”It’s [the project] moving more and more specific to AWS, yes. I think we could still deploy our stack on other cloud environments if we needed to, maybe this is something very specific to what we’re doing here, but we’re just finding more and more of our clients have a preference for AWS.”	’AWS’
F46	”The thing that we found was that Sumologic was very good at doing logging but was not very good at doing metrics. Whereas Datadog was very good at doing metrics and wasn’t so good at doing logging. However, three years on, Datadog is now very good at doing logging and the other tool I haven’t even talked about in quite a while. So, I think that the space of monitoring has changed in these recent years, and those tools have just gotten better.”	’State of the Art’, ’Reason for Specific Tool’
F47	”That big that project I mentioned before that had lots of EC2 instances and over-provisioned databases, we were using Datadog for metrics and logging as well.”	’EC2’, ’Virtual Machines’, ’Over-Provisioning’, ’Database’, ’3rd Party Monitoring Tool’, ’Datadog’
F48	”All that was just in this one tool, whereas now on this new project we’re working on, which is all serverless, we don’t have any external monitoring solutions. We’re all just using the AWS ones, because they have also matured to a point where we think it’s pretty good what we get now.”	’Serverless’, ’Provider Native Monitoring Tool’, ’AWS’, ’Reason for Specific Tool’
F49	”If the metric is timely and accurate [it is most useful]. Maybe that sounds obvious, but you can get metrics that are wrong. I actually have an example from today with this issue that I’m debugging and some of the memory metrics that I’m getting don’t seem to align with reality”	’Metrics’
F50	”And that they’re [metrics] timely, if you are only able to visualize your metrics once every 10 minutes or 20 minutes or so, then that’s not always detailed enough. Sometimes you want to respond to metrics, for instance, if you’re scaling based on CPU usage, which is a common pattern to do.”	’Metrics’, ’CPU Metric’, ’Auto-Scaling’
F51	”If your CPU suddenly is 100% for five minutes, you probably need some more servers to handle the load. Once that utilization goes down, then you know you can turn off a bunch of instances, so having that data as speedily as possible is very important.”	’CPU Metric’, ’Auto-Scaling’
F52	”Going back to the accuracy, on the AWS side, when you look at databases there are different metrics that you can get for memory usage, actually some of them are not exactly what they say they are. There are multiple memory metrics that you get, and if you don’t know the differences between them, you could easily make the wrong scaling decisions based on those metrics. Memory is known for being a very difficult thing to get right, so you can’t always trust those metrics, but you really need to know your stuff on that. So, I think that’s a very important thing, to me at least, metrics need to be usable.”	’Memory Metric’, ’Developer’

F53	”The new project that I’m working for is very different because it is ingesting large amounts of data in real time. And we’re making scaling decisions based on that because it’s serverless. Our performance is very heavily tied to the data we have coming in. Obviously, I can’t talk too much about the project, but when Black Friday comes around and there’s lots of financial transactions, our system is going to get spammed with different requests and we have to respond to that. There’ll be a lot more financial transactions that happen on that day. It matters immensely that we’re collecting that data and can make informed decisions.”	’Serverless’, ’Monitoring Reason’, ’Auto-Scaling’, ’Project Constraints’
F54	”Also, we’re dealing with data that is in queues and if those queues get full, we start to miss messages. And if we start missing messages or dropping them, then we’ve got problems because then we lose our integrity. Due to the nature of the product that we’re building that would cause all sorts of different problems. So, we need to always have very accurate metrics on how full our queues are, if we’re behind, if we can consume messages at the rate of which we are receiving them. We need to know all of that so that our application is healthy.”	’Metrics’, ’Monitoring Reason’, ’Project Constraints’, ’Queue’
F55	”Well, the more visual the better. Visual graphs. Well, if cost is a concern and you need management on your side, having visualizations is very useful because a graph can sometimes make an argument a lot better than humans can. So, if you show a graph that’s spikes up, it’s very obvious that something’s wrong. So yeah, I think those sorts of visualizations matter.”	’Graph’, ’Cost Monitoring’
F56	”But also, graphs can be very easy to misinterpret, which is true in all facets of life. But I know of graphs that have been misinterpreted because people haven’t read the scale correctly and a graph that sort of jumps up and down, might actually just be a tiny little blip. It’s just that it’s zoomed in so far that you only see a little bit.”	’Graph’, ’Developer’
F57	”If you had asked me a couple years ago [what is lacking in the currently used monitoring tools], I would have had a very long list for you on AWS of things that are missing, but there’s been a lot of improvements there.”	’Lacking in Monitoring’, ’Provider Native Monitoring Tool’, ’AWS’
F58	”As I said I think tracking memory is sort of difficult and I think sometimes just the overwhelming number of metrics and things that are available can be difficult. Which goes back to the theme that I’ve touched on: The difficulty sometimes is in interpreting things.”	’Memory Metric’, ’Developer’
F59	”I think we have most of the metrics available to us that we need, at least for the sort of projects that we’re working on and the things that we’re building.”	’Lacking in Monitoring’, ’Metrics’
F60	”It’s just the configuring of them [monitoring tools] and sometimes the cost of monitoring can also be very high. I know in our current project about 1/5th to 1/4th of our bill is monitoring and logging. Which is actually quite a lot.”	’Cost’, ’Monitoring’
F61	”Yeah, I think it [a waste monitoring tool] would be useful because it’s very easy to over-provision things. Even working in an environment where you’re told cost doesn’t matter, if you save money, and you are able to show that you’ve saved money, that is always appreciated, and it’s always noticed.”	’Value of Waste Monitoring’, ’Over-Provisioning’, ’Project Management’
F62	”We have talked about waste and cost, and I always think of the environmental impact of this. It’s stupid to spend insane amounts of money on infrastructure that is just going to destroy the planet and is just sitting there idle, not doing anything. When you look at the amount of energy consumption on all these data centers, it’s crazy. Once you start thinking in terms of what the environmental impact of running these things is.”	’Environmental Impact’
F63	”I think that in cloud we don’t always think about it in those terms. Like how much CO2 are we letting off by actually running this instance? Because don’t see the CO2 emissions, you just see the costs, and if you’re told cost doesn’t matter, that is just not how things should be. You should have a more holistic view of that and see the environmental impact of these things.”	’Environmental Impact’
F64	”I think it [a waste monitoring tool] would [be useful], especially if it does put it in those environmental terms. I think that makes it more tangible than just money. Sometimes business people only care about money and they don’t care about the environment. But I think that would be really useful.”	’Value of Waste Monitoring’, ’Value of Waste Monitoring’, ’Environmental Impact’, ’Project Management’
F65	”A lot of those metrics and tools are available to you in AWS and there are lots of tools to benchmark these sorts of things. However, there is not always the drive from organizations and from management to really maximize the cost and performance of the applications. I really think it should come from the top and it just doesn’t a lot of the time. I think a lot of the time it’s all about just getting things running in the cloud.”	’Provider Native Monitoring Tool’, ’AWS’, ’Metrics’, ’Project Management’
F66	”I know that the projects that I’ve worked on have been greenfield projects, so new ones where we’re building something fresh in the cloud. But for organizations that are migrating to the cloud, from on-premise, just doing this kind of lift and shift thing. From my experience, they’re always the worst in terms of utilization because they’re usually over-provisioned in the data center and then that sort of gets carried with it into the cloud. Cloud actually offers so many ways to be able to monitor the performance and optimize it, but there’s not always that drive to do that. It’s purely going to cloud so there’s a missed opportunity there for a lot of companies, I think.”	’Project Type’, ’Over-Provisioning’, ’Optimization’, ’Monitoring Engagement’
F67	”I’m not sure about [how waste monitoring could be useful in] CI/CD, I might be able to see how that could work.”	’How to Waste Monitor’, ’CI/CD’

F68	”I don’t think it [a waste monitoring tool] would be a run once and check it because then it becomes just a box that you tick. Like yeah, where environmentally friendly, we ticked that box.”	’How to Waste Monitor’, ’Environmental Impact’, ’Project Management’
F69	”I think something that runs continually would actually be really cool. I’m a very visual person so I like dashboards and graphs and things like that. So, if you had something running in real time that showed you your current sort of utilization of all of your resources and that that could give insights into what you could scale down. Yeah, I think that would be really useful. So continuously available for sure.”	’Dashboard’, ’How to Waste Monitor’, ’Graph’, ’Metrics’
F70	”I know that AWS, at least on paper, they say that they will have 100% renewable energy in their data centers by 2025. Which is 3 years away, so who knows whether they’ll meet those targets. If the experience of governments and meeting environmental targets is anything to go by, they probably won’t meet or come anywhere near meeting any of their goals. But we’ll see, I’m very curious to see what happens. I think that any organization or company that has any thought or care about the planet that we live on should definitely be taking this sort of stuff seriously.”	’Environmental Impact’, ’Project Management’, ’AWS’
F71	”And I like that AWS has added sustainability as one of their pillars of the AWS Well-Architected framework. I don’t hold AWS up on a pedestal of good governance because I think they do a lot of things that are appalling. But it’s good to see that they are, at least on paper, talking about cloud waste and environmental impact.”	’AWS’, ’Environmental Impact’
F72	”To give you an anecdote from this project I’m working on at the moment, where we’re ingesting this mass amounts of data, where we’re queuing it, we’re using Lambda, we’re using DynamoDB. For development purposes, we’ve actually set up like 8 copies of our infrastructure. Where we initially thought it might blow out the costs, but at least it’s serverless so we’re only paying for what we use. So, it shouldn’t be very much. We’ve been working on it for all of these months, and we checked the bill the other day to see how much we used. And we were at, I think €44 for the month.”	’Lambda’, ’Queue’, ’DynamoDB’, ’Managed Services’, ’Serverless Functions’, ’Cost’
F73	”So, we had this whole conversation about these Lambda scripts that we were running to generate load, like hundreds of thousands of messages. We’re starting off small because it’s going to be millions. Someone was really worried about how much this was going to cost, and we looked at the bill and it was €0,13. We only pay for when we use it, so we’re not paying any idle costs for our infrastructure, which I think is really great.”	’Lambda’, ’Serverless Functions’, ’Cost’
F74	”Does that mean that the servers in the data center are turned off when we’re not using them? No, so I’m curious to see what this does from an environmental impact perspective. What the relationship is between those things because your bill might be less, but it might not make a difference for the CO2 emissions from the data center. That would be really interesting to dive into but that’s more specific to how the data centers work.”	’Environmental Impact’

C Questionnaire

Waste monitoring feedback questionnaire

The following are some questions about your background. The answers will be used to characterize the demographics of the experts. Your name will never be mentioned in the thesis, so the data will be anonymous.

1. What is the highest level you completed (HBO/WO Bachelor/Master/other)?

2. Which study program did you do?

3. What is your role in Deloitte?

4. How long have you been fulfilling this role?

5. How many years of experience do you have in Cloud Engineering or similar fields?

6. How many Cloud Engineering related projects have you been involved with during those years in Deloitte?

Waste definition

Waste is defined as a ratio of the resources that are not being utilized to a preferred threshold and is bound to 0. A simple example is memory being utilized at 60%, if the threshold would be set at a 100% the waste ratio will be 40%. This works for any factor of which you have utilization metrics available (e.g. vCPU, memory, storage, IOPS, etc.). The factors can be combined to get the waste ratio for a certain instance or a certain service. Binding the waste ratio to 0 is to keep it from getting negative values if the threshold is passed.

1. On a scale from 1 to 5, how useful is the waste ratio metric as defined above in your opinion?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

2. On a scale from 1 to 5, how useful is it to have the waste ratio metric available on a service, instance, and factor level? (with factor we mean a specific aspect of an instance for which the utilization can be measured, e.g. vCPU, memory, etc.)
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

3. What do you think overall about this definition of waste?

Waste threshold

Since it is undesirable to have a, for example CPU, run at a 100% utilization, the preferred utilization threshold can be changed. The default threshold will be based on the auto-scaling rule for scaling out, when available. For example, if a CPU has an auto-scaling rule for scaling out with a threshold set at 80% or above utilization for a set amount of time, the threshold will be set to 80% as well. If the CPU is running at a utilization of 60% for a period of time the waste is only 20%, since it is undesirable to go above the 80%. If there is no auto-scaling rule for scaling out available the default threshold will be 100% and should be changed manually.

1. On a scale from 1 to 5, how useful do you think the ability to manually change the threshold is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

2. On a scale from 1 to 5, how useful do you think basing the default threshold on the auto-scaling rule for scaling out is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

3. Do you have other suggestions on how the default threshold can be determined?

4. What do you think about the threshold used in the waste ratio calculation?

Waste factors

A service usually has multiple factors that influence the waste metric. A compute service usually has CPU, memory, and sometimes GPU utilization metrics. In order to calculate a waste ratio for the instance as a whole a weighted average is used. Each factor gets a default weight, which can be changed. For instance, if you use a memory optimized compute instance, the weight for memory waste will be considerably higher than the weight for CPU waste.

1. On a scale from 1 to 5, how useful do you think using weights for each factor is in calculating the waste ratio of a whole instance?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

2. On a scale from 1 to 5, how useful do you think having default weights based on the service and instance type is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

3. What do you think about the use of weights for each factor depending on the instance type?

Potential savings

The waste ratio can show how much waste is present in an application, service, or instance, but in order to easily see how much of an impact it has on the costs of the resources for the application it needs to be converted to a monetary value. Because the price does not scale linearly with the size of an instance, this is only an estimate, which is why I call it “potential savings”. The potential savings are based on the average waste ratio over a certain time frame and the price of the resource during that time.

1. On a scale from 1 to 5, how useful do you think converting the waste ratio to a monetary value is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

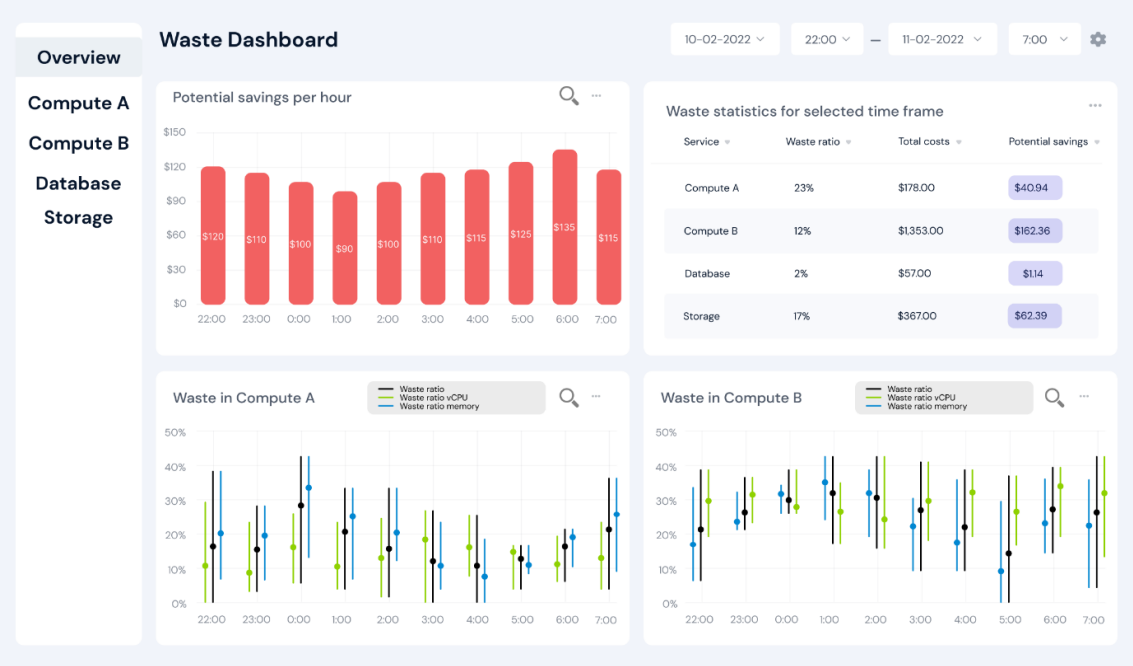
2. On a scale from 1 to 5, how useful do you think the potential savings metric is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

3. What do you think about converting the waste ratio to a monetary value?

Waste Monitoring Dashboard

Since all the information above should be presented in a dashboard, I created mockups of such a dashboard. The layout is of no importance here, seeing as the tool should be integrated with the monitoring tools that are already being used. The information that is being presented and the way it is presented is important.

Waste Monitoring Dashboard – Application Overview



Above is the general overview of the waste dashboard. On the left are buttons for the general overview of all services and for each service that is in use. On the top right you can set the time frame that you want to see. Note that all numbers are just for the mockup and do not necessarily correlate to each other correctly.

The top left box is for the total potential savings, this shows the wasted costs for the entire application based on the different waste ratios and the actual costs of running the services in the selected time frame. The potential savings are calculated per hour in this example, depending on the selected time frame the potential savings can be per multiple hours or even days. The total potential savings over the time frame is the sum of all the data points. This sum is shown in the table to the right of the graph. This table contains the waste ratio, total costs, and potential savings for each service over the time frame selected. The bottom two boxes show the waste ratio for specific services in the application. You can scroll down to show the other services that do not fit on the screen. If you click the magnifying glass icon you get a more detailed overview of that specific service (you can also use the buttons on the left to do this).

1. On a scale from 1 to 5, how understandable do you think the presented information from this view of the dashboard is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

2. On a scale from 1 to 5, how useful do you think the information about potential savings presented in the graph on the top left is?

1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

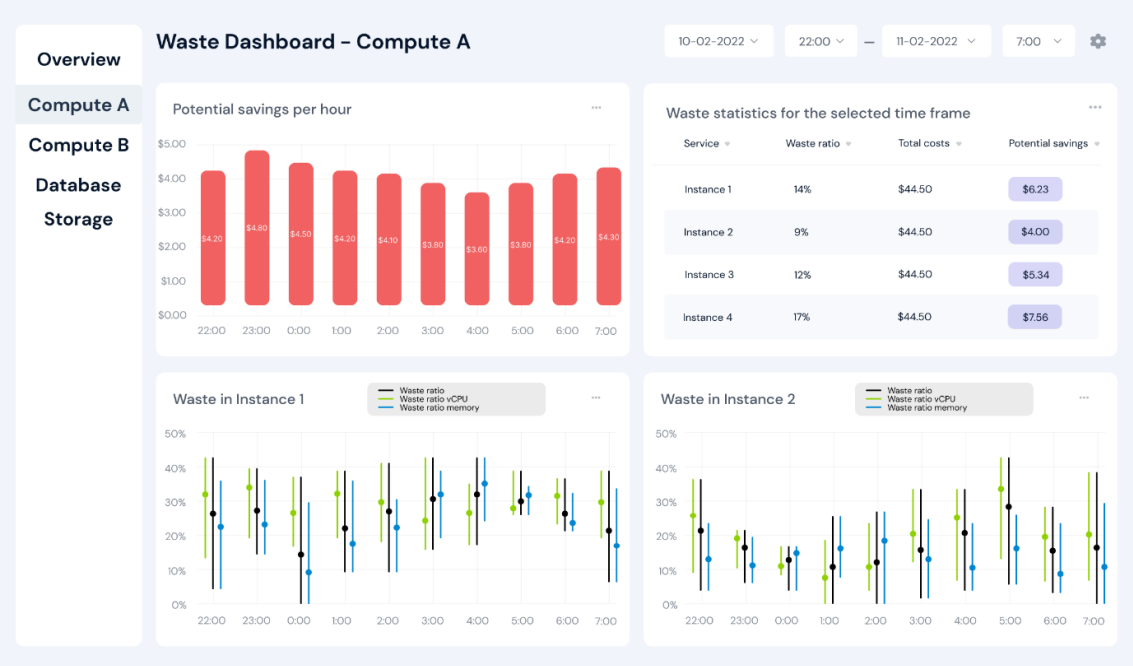
3. On a scale from 1 to 5, how useful do you think being able to see the potential savings over time is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

4. On a scale from 1 to 5, how useful do you think the table showing the data over the entire time frame is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

5. On a scale from 1 to 5, how useful do you think being able to see the waste ratio per service over time is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

6. What do you think about the general overview of the waste monitoring dashboard? (Please note that the actual layout is of little importance for the scope of this project)

Waste Monitoring Dashboard – Service Overview



Above is the mockup of the waste dashboard for a specific service. This is very similar to the general overview, but now gives information about the specific service instead of the whole application. In the table on the right it shows the waste ratio, total costs, and potential savings for each instance over the selected time frame. The bottom two graphs show the waste ratio for specific instances over time. If you scroll down you can see the other instances as well.

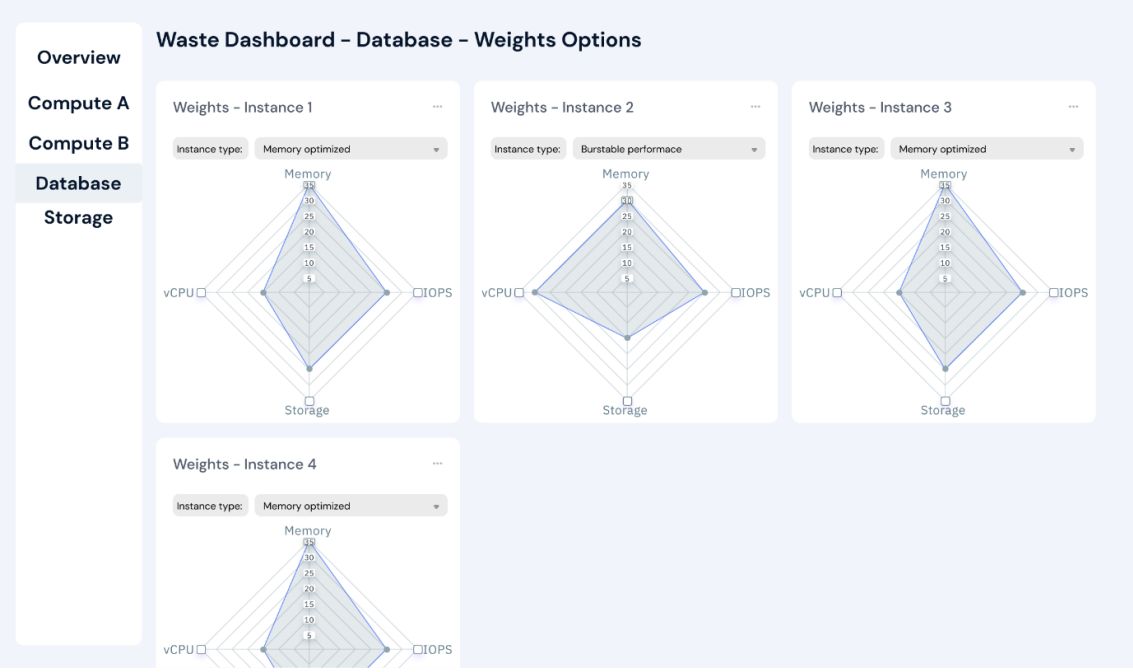
1. On a scale from 1 to 5, how useful do you think being able to get a more detailed overview for a specific service is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

2. On a scale from 1 to 5, how understandable do you think the information presented about specific instances is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

3. On a scale from 1 to 5, how useful do you think the information presented about specific instances is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

4. What do you think about the overview for a specific service in the waste monitoring dashboard? (Please note that the actual layout is of little importance for the scope of this project)

Waste Monitoring Dashboard – Weights Options View



The above mockup is for changing the weights in the waste ratio calculation (as explained earlier). Depending on the service that is being viewed the factors (e.g. vCPU, memory, IOPS, etc.) that need a weight assigned are different. Depending on the service and the instance type different default weights are used. This menu lets you change these weights to be more specific for the use case at hand.

1. On a scale from 1 to 5, how useful do you think being able to change the weights manually using a chart is?
   1 ☐ 2 ☐ 3 ☐ 4 ☐ 5☐

2. What do you think about the weights options menu of the waste monitoring dashboard?

Lastly, is there anything in general that you would like to add?