Ing. Patricie Suppala

FinOps in SaaS platform within hybrid, multi-cloud, multi-tenant, multi-region environments

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

2 April 2022

Abstract

Author:	Ing. Patricie Suppala
Title:	FinOps in SaaS platform within hybrid, multi-cloud, multi-tenant, multi-region environments
Number of Pages:	66 pages
Date:	2 April 2022
Degree:	Bachelor of Engineering
Degree Programme:	Information Technology
Professional Major:	Mobile Solutions
Supervisors:	Tommi Olavi Lundell, R&D Squad Group Lead Anand Adhiappan, Head of Advanced Solutions Erik Pätynen, M.Sc., Senior Lecturer

---

Throughout the past decade, Software as a Service (SaaS) has been growing significantly. As traditional industries are preparing an entrance to SaaS, they must consider which approaches to adopt. Transfer to SaaS includes adopting tools and processes for managing expenses, including aspects of Cloud Financial Operations (FinOps), which aims to ensure a return on every spend in the public cloud. Managing expenses also means going beyond FinOps and striving for cost-effective SaaS offerings and cost optimisation on a functionality level.

This project researches the environment and tools that could be used for FinOps and other SaaS cost optimisations. It strives to answer the following questions:

- What are the required capabilities for managing costs?
- What are the major cost contributors?
- What are the possible solutions?

Solutions should be suitable for hybrid cloud, multi-cloud, multi-tenant and multi-region environments. This project has been conducted with research performed for an international company, a major leader in its industry.

As a result of this research, three main solution types have been identified: commercial, public cloud-native and open source. The open-source solution was investigated in great detail and a toolchain has been proposed based solely on open-source tools. The main pitfalls of open source have been highlighted. It is also suggested that commercial solutions be taken into consideration.

Keywords: Cloud Computing, Software as a Service, Public Cloud, Costs, Multi-cloud, Hybrid Cloud, Multi-Tenant, KPI, FinOps

Contents

Contents

List of Terms and Abbreviations

FinOps	Cloud Financial Operations
ASP	Application Service Provider
VPN	virtual private network
SLA	service quality agreement
LAN	local area network
AWS	Amazon Web Services
ECS	Amazon Elastic Container Service
GKE	Google Kubernetes Engine
OCP	Red Hat OpenShift Container Platform
ACK	Alibaba Cloud Container Service for Kubernetes
AKS	Azure Kubernetes Service
SOA	service-oriented architecture
MOA	microservice-oriented architecture
ESB	enterprise service bus
API	application programming interface
KPI	key performance indicator
B2B	business to business

MRR monthly recurring revenue

ARPU average revenue per user

CLTV customer lifetime value

CNCF Cloud Native Computing Foundation

JDBC Java Database Connectivity

ICMP Internet Control Message Protocol

TCP Transmission Control Protocol

ETL extract, transform, load

EMG Elasticsearch, Metricbeat, Grafana stack

LOC lines of code (LOC)

TBM Technology Business Management

Hybrid cloud – combination of private and public cloud.

Multi-cloud – combination of several public clouds. For example, some services may be running on AWS. The same services for the same customer may run on Google Cloud, depending on the availability to deliver the service under contracted conditions. The ability to work in multi-cloud is especially relevant for services running on the edge and in multiple regions. [1]

Multi-tenant – discussed in detail in Chapter 2.

Multi-region – one service running in multiple regions [2]

SaaS Software as a Service

CAPEX	capital expenditures
OPEX	operating expenses
ERP	enterprise resource planning

1 Introduction

A shift towards Software as a Service (SaaS) can be seen across industries. Pricing models, deployment, and business operation models are changing to allow customers more flexible use of services and offer additional features which cannot be available on-premise. Due to its numerous advantages, it is anticipated that many of these services will be managed and delivered using Kubernetes-orchestrated environments. As a result, infrastructure for these services will be entirely abstracted. Before this becomes a reality, many vital challenges remain. These challenges include metering, allocating, and calculating costs related to these services for cost management and billing.

This thesis aims to analyse options for cost-related metering per service usage, preferably using open-source tools. Ideally, the solution would be a modular, flexible toolchain for all possible payment models and applications that is usable across multiple public cloud providers.

SaaS products are marketed as more economical. Therefore, managing costs is a significant concern for development companies, who need to ensure that their affordable products are also profitable.

This research has been performed on demand for an international company, a major leader in its industry. While conducting this research, the company was also running other parallel research and initialising pilot projects. The company’s market is worldwide in multiple industries with a high level of competition and a strong focus on cost efficiency. The competition includes major established players of comparable size, market disruptors, and innovative customers in the supply chain. The concept of SaaS was considered at the start of the project. A fast-paced and high-stakes working environment could not be

successful without everyone’s willingness to quickly evaluate and re-evaluate any options and possibilities.

Beyond rapid changes in the understanding of related matters within the company, SaaS options are also changing rapidly. Although some actors might be considered established in the commercial sphere and many more are emerging, when concentrating on open-source possibilities, the options are not stable. Although new and interesting open-source tools continue emerging, the dynamics of open source itself cause some communities to grow quickly and others to slowly fade away. All of this must be weighed when seeking suitable tools and solutions. Therefore, also this thesis reflects the current best possible understanding of relevant matters at the time it was written.

2 Overview of the current SaaS monitoring landscape

SaaS is undergoing rapid development, and terminology may carry different meanings depending on industry or application. The purpose of the following chapters is to define terminology and context and explain SaaS and cloud computing broadly.

2.1 SaaS history and definition

SaaS history started in the 1990s with the gradual centralisation of enterprise servers. Businesses discovered that they could achieve savings by centralising computing resources. Another supporting factor was the proliferation of the internet, which is based on the client/server model. However, there were no changes in the ownership or the way of managing application resources. [3, 4]

The first SaaS-resembling model that provided application services was Application Service Provider (ASP)–hosted software with a client/server architecture. The client mainly used a virtual private network (VPN) via the internet. Businesses that used a vendor's ASP services connected to a dedicated server-specific application client or, later, to a more universally-used remote desktop. This model brought changes in licensing policy and ownership. The customer company no longer had to own the software and resources; they simply rented a service and paid for it at regular intervals. With the operator service, a service quality agreement (SLA) is established, which specifies application availability, the scope of support, rental prices, and other details. [4, 5]

A SaaS distribution model can be presented as an IT service with the goal of providing application functionality for the end customer. SaaS is defined with the following characteristics, presented by the SaaS Executive Council [6]:

- SaaS uses a one-to-many model or multi-tenant mode of operation. The same application environment serves more than one user. This architecture is the prerogative of the SaaS model.

- The application is managed centrally by the operator. The customer connects using a client that, in most cases, is based on web browsers.
- SaaS applications are based on the internet and its protocols. SaaS is closely tied to web application development.
- The application is usually data-bound. Similarly to ASP, the model assumes the storage of all data by the service provider.
- For SaaS, the application provider is also the manufacturer of the application.

The above definition is from 2006. Research shows that, in 2021, there is no one clear definition of SaaS. The SaaS model, unlike ASP, is not compatible with the model of perpetual license pricing. For the case of a service run on a private cloud for a single user based on a perpetual license or a subscription longer than one year, we would refer to managed service, not SaaS. [7]

2.2 SaaS driving forces

Sometimes SaaS is defined as a commercial model in which software is sold as a subscription rather than a perpetual license [8]. However, subscription model itself cannot define SaaS, as non-SaaS applications can also be billed as subscriptions [9], while SaaS applications can use completely different monetisation and billing strategies [10].

SaaS is a software product for which the developing company takes responsibility not only for application development and data management but also runtime, middleware, O/S, virtualisation, servers, storage, networking, maintenance, and so forth. In other words, all the customer has to do, for many services, is create a username and provide a valid e-mail address and credit card. This provides significant simplification and flexibility for the customer. [11]

There are several reasons why companies that develop software consider SaaS:

- For established companies, SaaS is another potential monetisation market – it offers the possibility of targeting users that cannot afford to buy a license for an on-premise solution.

- Less-customised software is gaining acceptance.
- SaaS has become popular even among major customers.
- SaaS is what the competition is investing in, and the concept has already been proven in some industries.
- SaaS is a way to offer solutions directly to customers without re-sellers claiming part of the profit.
- Cloud computing enables the introduction of new functionalities, for example, in areas of collaboration.
- SaaS does not have to consider legacy.

SaaS customers have their own motivations:

- SaaS solutions are new and offer a fresh look and feel; limited functionality at early stages of development adds to the simplicity. An interactive web interface allows for more intuitive navigation. Or, as some say, ‘SaaS is cool.’
- The current trend is to move from capital expenditures (CAPEX) to operating expenses (OPEX). This change brings value on several levels:
  - Customers are reaching out to OPEX to free capital for further investment.
  - Decisions can be made on the managerial level at which the tool is needed without permission from higher levels of the hierarchy.
  - No significant upfront investment is needed.
  - The bureaucracy required for large investments can be avoided.
- Currently, in some companies, the traditional review of terms and conditions can be bypassed, as processes are not set to prevent employees from accepting any terms and conditions while incorporation new SaaS/based tools into their daily work.
- The buying experience is smoother, as customers are spared the necessity of being pressured by salespeople for information about their use case and asking for demos and evaluation licenses. Evaluation can be conducted at one’s own time and pace.
- On-demand billing is an attractive alternative to subscriptions.
- Service level agreements are standardised and available online.
- Customers can still request custom development.
- SaaS is often accompanied by online documentation and a support community, which helps to lower the usage barrier significantly.
- SaaS is available from anywhere and for any device type and model.

- SaaS is scalable.
- SaaS offers to save costs, if not on the price of the product itself, then on the infrastructure and resources needed to deploy and maintain an on-premise solution.
- Upgrades are included in the price; that is, one always has the latest version.
- Fixes are deployed continually without service interruption, and customers do not need to wait for the next release.
- SaaS solutions tend to create a network of solutions offered by different companies that provide significant additional value. One example is the connection between Salesforce, ZenDesk, and Jira, through which even a small company can afford a full-fledged enterprise resource planning (ERP).
- Some SaaS solutions provide access to a plug-in marketplace to which anyone can contribute with add-on-type modular functionality.

Above is an extensive list of motivators for trying SaaS. It is thus fair to list the disadvantages and explain why companies leave SaaS and return to on-premise software. Among the most prominent disadvantages are the following:

- Costs that are potentially too high to maintain the same level of security.
- A lack of essential features that were present in on-premise solutions but have not yet been developed for SaaS versions.
- An extensive customisation and support come with additional costs.
- the loss of control over development (the interests of another customer with a different use-case may be prioritised).
- non-compliance with privacy laws; for example, Russian law forbids the storage of data about its citizens by datacentres outside of Russia.

2.3 Cloud computing

There are four deployment models recognised in cloud computing: the public cloud, private cloud, hybrid cloud, and community cloud.

Public cloud – This group of cloud services is publicly available to many users who represent end consumers or organisations. The use of public clouds almost always occurs via the internet.

Private cloud – A private cloud is a cloud computing environment dedicated to a single customer. As for public clouds, individual services can be available through the internet; however, private clouds often use a local area network (LAN) or another type of private network. [12]

Community cloud – This type of cloud solution expands the above-described private cloud. As its name suggests, the user base consists of users with common interests or shared concerns (such as security requirements or data privacy). In practice, this type of solution can be applied to different geographical areas or business communities, such as the BRIC community cloud, the community cloud of the European Union. [13]

Hybrid cloud – This final category represents a special cloud environment that encapsulates two or more other deployment models. This method is typically used by companies that already run a private cloud but need access to public services. [12]

A significant change was brought to SaaS by public cloud providers that offer the option to rent server capacity and accompany it by a portfolio of services that enable different functionalities, such as data storage, access, or processing. Figure 1 shows the leading public cloud providers and about 10% of the services they offer. It briefly illustrates the scale and competitiveness of this environment.

Category	Service	AWS	Microsoft Azure	Google Cloud	IBM Cloud	Oracle Cloud	Alibaba Cloud
Compute	Shared Web hosting		Azure shared App Services		Web hosting services		Web Hosting Simple Application Server
Compute	Virtual Server	Amazon EC2	Azure Virtual Machine	Compute Engine	Virtual Server Infrastructure (VSI)	Compute	Alibaba ECS
Compute	Bare Metal Server	Amazon EC2 Bare Metal Instance (Preview)	Azure Bare Metal Servers (Large Instance Only for SAP Hana)		Bare Metal Servers	Bare Metal Servers	ECS Bare Metal Instance
Compute	Virtual Dedicated Host	Amazon EC2 Dedicated Hosts		Sole Tenant Node (Beta)	Dedicated Virtual Servers Infrastructure (VSI)	Dedicated Compute Classic	Dedicated Host
Compute	Container Registration Service	Amazon EC2 Container Registry	Azure Container Registry	Container Registry	IBM Cloud Container Registry	Oracle Cloud Infrastructure Registry	Container Registry
Compute	Container Management Service	Amazon EC2 Container Service Amazon Elastic Container Service for Kubernetes (EKS)	Azure Kubernetes Service (AKS) Azure Container Instances	Kubernetes Engine	IBM Cloud Kubernetes Service	Container Engine for Kubernetes (OKE)	Container Service Container Service for Kubernetes
Compute	Micro Services App Development Platform	AWS Lambda	Azure Service Fabric Azure Functions Event Grid	Google Cloud Functions	IBM Cloud Functions	Oracle Functions	Function Compute
Compute	Virtual Private Servers	Amazon Lightsail	Azure App Service Environment		Virtual Server Infrastructure (VSI)		Simple Application Server
Compute	Auto Scaling	Auto Scaling	Azure Autoscale Virtual Machine Scale Sets	Auto Scaler	Auto Scaling	Auto Scaling	Auto Scaling
Compute	Batch Jobs	AWS Batch	Azure Batch	Preemptible VMs	Workload Scheduler		Batch Compute
Compute	App Development/ Deployment (Java /.Net /PHP /Python)	AWS Elastic Beanstalk	Azure Web Apps Azure Cloud Services	Google App engine	Cloud Foundry Apps	Application Container Cloud Java Cloud Service	Enterprise Distributed Application Service
Compute	Event Driven Computing	AWS Lambda	Azure Functions Event Grid	Cloud Functions	IBM Cloud Functions	Oracle Functions	Function Compute
Storage	Object Storage	Amazon Simple Storage Service (S3)	Azure Blob Storage	Cloud Storage	Cloud Object Storage	Object Storage	Object Storage Service
Storage	Virtual Machine Disk Storage	Amazon Elastic Block Storage (EBS)	Azure Page Blobs / Premium Storage Managed Disks	Persistent Disk	Block Storage	Block Storage	Block Storage
Storage	File Storage (SMB Compatible)	Amazon Elastic File System (EFS)	Azure File Storage	File Store	File Storage	Oracle Cloud Infrastructure File Storage	NAS File Storage
Storage	Long Term Cold Storage	Amazon Glacier	Azure Archive Storage Azure Cool Storage	Cloud Storage Archival Storage (NEARLINE & COLDLINE)	Object Storage-ColdVault	Archive Storage	Object Storage Archive
Storage	Hybrid Storage/Storage Gateway	AWS Storage Gateway	Azure Storsimple			Storage Software Appliance	Hybrid Cloud Storage Array

Figure 1 Public cloud services by public cloud provider [14]

2.4 Kubernetes

Kubernetes is an open-source tool that allows docker orchestration. Another tool offering similar functionality is , for example, Docker Swarm. However, Kubernetes has managed to gain significant traction. It was adopted by all major public cloud providers (AWS, Google Cloud, RedHat, Microsoft Azure) in their commercial versions: Amazon Elastic Container Service (Amazon ECS), Mirantis Kubernetes Engine (former UCP), Google Kubernetes Engine (GKE), Red Hat OpenShift Container Platform (OCP), Alibaba Cloud Container Service for Kubernetes (ACK) and Azure Kubernetes Service (AKS). [15, 16, 17, 18, 19, 20, 21, 22, 23]

2.5 Monitoring

Monitoring refers to the process of collecting, storing, processing, analysing, and visualising data obtained with a monitoring system. The goal is to create a picture of the current state of the monitored environment, with the additional aims of forecasting the system's future behaviours, anticipating the behaviour of similar systems, or evaluating a system's behaviour against its past. The next chapter will show that when monitoring costs in SaaS, it is desirable to have a granularity, at least on the microservice level.

2.6 Alerts

Sending alerts is the aspect of a monitoring system responsible for carrying out actions based on changes in measured values. An alert definition always consists of two components:

- Threshold – A threshold is the value above which an action is performed.
- Action – An action is performed after exceeding the given threshold value. An action is usually sent to predefined employees or user groups and contains information about where and when the event arises. [24]

The primary purpose of an alert is to warn of a change in the status of monitored systems, usually to notify a human for further evaluation. Alerts are triggered

when a metric exceeds a predefined or calculated threshold. Thresholds can be based on past values or they can be forecast. Thresholds can be defined manually or with the help of machine learning.

2.7 Architectural styles

The gradual evolution of programming paradigms in information systems began by promoting object-oriented languages instead of procedural languages. However, object/oriented programming on its own could not solve all problems, and applications continued to grow to the point at which they were too large and difficult to maintain. Moreover, any changes made to one module affected the behaviour of other parts of the system, and testing thus took much time. For this reason, architectural styles appeared, especially service-oriented architecture (SOA) and microservice-oriented architecture (MOA).

SOA first appeared in 2000. The motivation for this approach was the need for more flexible architecture and configuration depending on clients' needs. SOA can therefore be described as an architectural model in which individual components offer services to other components through communication protocols within a computer network. This solution's advantages lie in interconnecting the various parts of an application, its central administration, and the reusability of already-established services. Enterprise service bus (ESB) mediates communication between all participants. However, due to using a central control point, the system is also more prone to failure. For example, if one service is slow, it will affect other system parts.

MOA brings more power and less fragmentation to the system. It is typically used, for example, in banking applications. It is different from SOA in that it shares databases across all services. Communication occurs through messaging, the format of which depends on the specific implementation; but XML is used most often. The most important part of the service is its application interface (API), through which interactions with other parts of the system occur. Whether the communication between services is asynchronous or synchronous,

as well as other aspects of communication, depends on the specific case. Most often, HTTP is used as a communication protocol together with the REST architectural model. The format for data exchange is, in most cases, JSON. Regarding the mutual communication of microservices, there is a delay in sending messages over the network. [25, 26, 27]

2.8 Microservices in Kubernetes

Microservice is one way of offering SaaS. It is an architectural style, as discussed in the previous section. Microservices together create what would in traditional computing be called an ‘application’. For example, Netflix consists of around 1000 microservices, including movie encoding [28]. What seems from the customer side to be one service is, from the perspective of the service provider, a service mesh, layer, or architecture that facilitates communication between microservices. [29, 30]

Figure 2 shows a simple service that consists of only one microservice running on Kubernetes on three pods simultaneously. The traffic to these pods is managed with a load balancer.

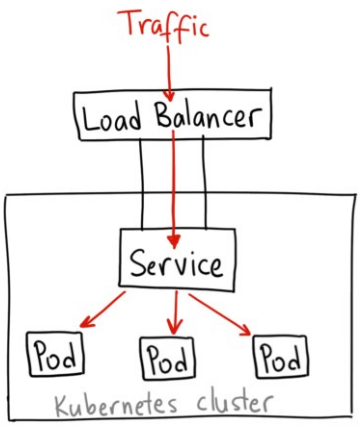

Figure 2 One service consisting of one microservice and running on three pods [31]

An example of a service consisting of two microservices and running on two pods, with Ingress providing external access and load balancing, can be seen in Figure 3.

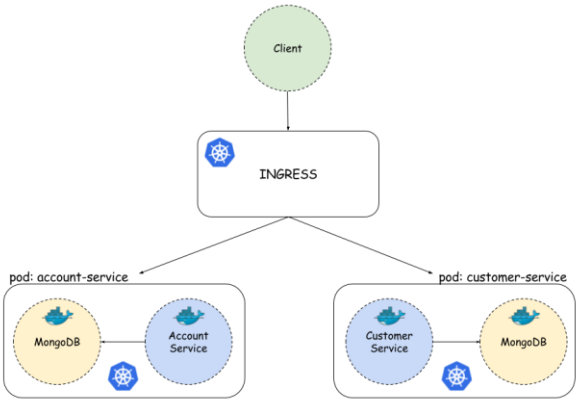

Figure 3 One service consisting of two microservices and running on two pods [32]

The matter becomes more complex when adding microservices, combining services, and running services in multiple clusters or over multiple regions. Figure 4 shows a very simplified example of such a scenario. The tools required to make this possible will be discussed in Chapter 3.

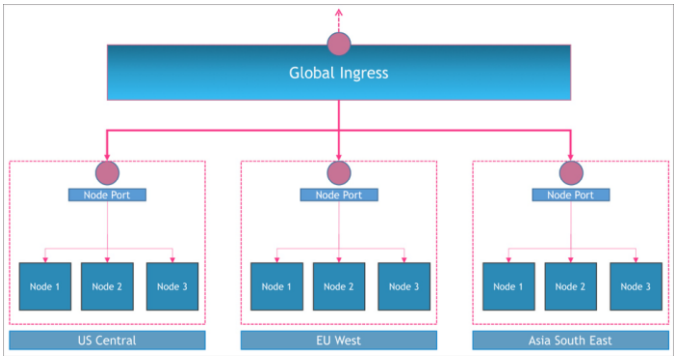

Figure 4 Managing multiple regions with Ingress [33]

To limit the possible number of combinations between service, region, latency, and any other aspects of the service level agreement, the service provider may choose to predefine viable scenarios and make service available in a service

catalogue. A customer's selection is then deployed using the service broker, which again contains a predefined guide on how to deploy a given scenario. Figure 5 shows the high level architecture of a service broker that provides services to an application based on a service catalogue.

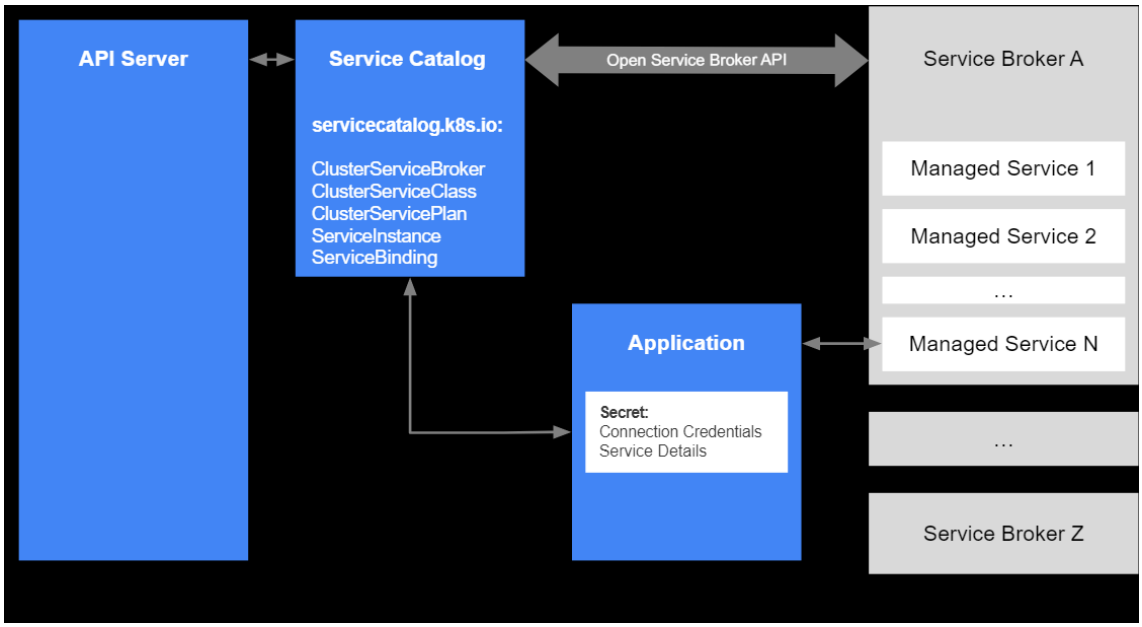

Figure 5 Offering and deploying services with a service catalogue and service broker [34]

2.9 Single-tenant vs multi-tenant

When providing software as a service, there are two basic approaches: single-tenant and multi-tenant. These terms define the number of tenants using one instance of the application.

In the single-tenant model, a custom application instance exists for each tenant. Each of these copies is then run on a separate infrastructure, shielded from other systems. In practice, this is most often the provider's server or individual hosting of the relevant tenant. All instances, for example, contain an independent database and possibly other supporting tools. This model is shown in Figure 6, which illustrates, in a simplified way, the challenge of defining multi-

tenancy. A fourth scenario is missing from figure 6, specifically, a single-tenant application with a multi-tenant database.

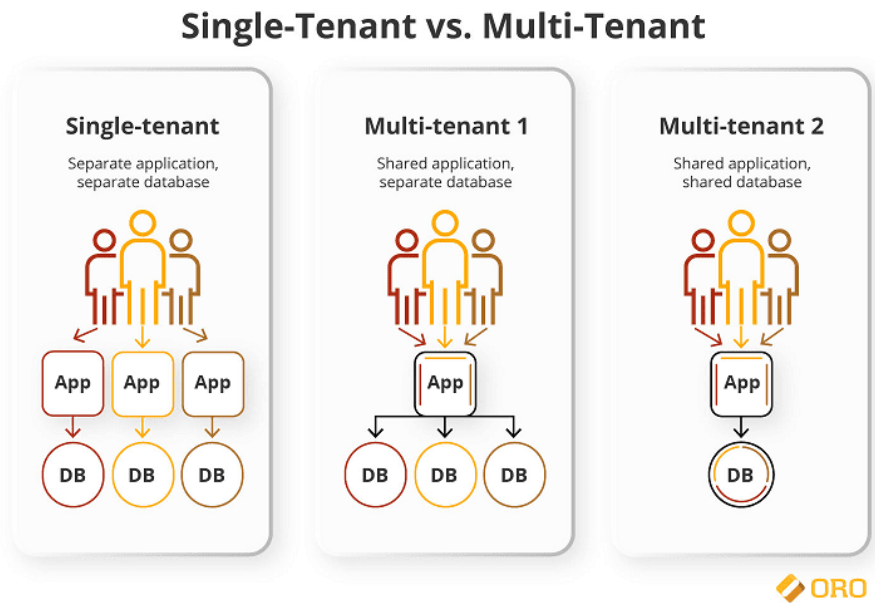

Figure 6 Single-tenant vs multi-tenant [35]

The multi-tenant model, in this case, means that one instance of the application serves all of the provider's customers. Such a system is operated centrally and contains some shared parts, for example, administration. At the same time, however, every tenant has its own views or modules visible only to its users and not available to other parts of the system. All clients can also use settings to customise the application to their preferences. Leased systems typically include user management or the option to add or remove tables in the database.

2.10 Key performance indicators

Key performance indicators (KPIs) are metrics used to evaluate a system over a period of time and plan its future behaviour. The selection of SaaS-specific KPIs depends on a company's business, cost, and pricing models. For example, a

B2C service with a pure consumption character and tier pricing will require different KPIs than a B2B service that creates business dependency and requires support services to onboard or to leave, or one priced based on usage. An overview of the most commonly cited cost-related KPIs is below. [36, 37]

Monthly recurring revenue

Monthly recurring revenue (MRR) is one of the most significant indicators for companies using a subscription business model, a model based on regular payments from customers who pay the (usually monthly or annual) subscription fee. MRR represents the income that the company can depend on each month. Changes in customer structure (newly acquired. and lost customers) must be addressed when evaluating MRR.

Churn rate

Churn Rate determines the percentage of customers who are lost in each period (cancelled subscriptions). If a company has ten subscribers and one customer has cancelled their subscription in the last month, the churn rate is 10%. The churn rate indicates customer dissatisfaction with the company.

Average revenue per user

Average revenue per user (ARPU) indicates the company’s average income from one customer. The formula divides the total revenue for a certain period by the number of subscribers in the same period. To increase the ARPU, companies can upsell (offer a more expensive premium product) or cross-sell (offer a complementary product).

Customer lifetime value

Customer lifetime value (CLTV) measures the revenue that a customer will bring to the company for the entire period of trading with the company. The funds spent on acquiring new customers should be smaller than CLTV. CLTV is

a prediction; it is not an exact number for a specific customer group. CLTV shows the revenue that can be expected when acquiring a new customer. This expectation is based on past data.

Quick ratio

The quick ratio is an indicator of short-term liquidity. It measures the ability to cover short-term liabilities with the most liquid assets. In the SaaS environment, the quick ratio is the ratio of revenue growth to losses. The quick ratio provides a view into how much and how fast the company is growing. For a company to prosper, the quick ratio must be kept above 1. A quick ratio above 4 means that a company is growing quite quickly and efficiently, making it easier to attract investors.

2.11 Open-source

When considering open source for a large-scale commercial project by a company that is not one of the major open-source contributors, it is important to consider the following issues:

- The viability of the open-source tools and how likely they are to persist in the future. If a critical open-source component loses traction or ceases to be open source, the company would need to take over its development, whether as open-source or as a fork (if the license enables it).

- The license of the open-source solution. Some licenses may require that parts of code are also open source, that the code of the solution must be available to the customer, or even that the final product must be open source.

Compared to in-house development, using open source involves a loss of power over the final product. This power ultimately goes to the companies actively contributing to (paying for) the development of the open-source product. Information about which companies are currently in charge of open-source

projects is not readily available. This information can be inferred from the Cloud Native Computing Foundation website by analysing the foundation project partners or the contributors' e-mail addresses. [38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50]

3 Tools

The following subchapters describe some of the available tools and solutions that can monitor cost-related metrics in SaaS. With the overview in Table 1, each project's size and future potential can be quickly compared with others.

Tool	Website GitHub	Year started	Contributors
ElasticSearch	https://www.elastic.co/elasticsearch/ https://github.com/elastic/elasticsearch	2010	1641
Telegraf	https://www.influxdata.com/time-series-platform/telegraf/ https://github.com/influxdata/telegraf	2015	887
Prometheus	https://prometheus.io/ https://github.com/prometheus/prometheus	2012	593
Logstash	https://www.elastic.co/logstash/ https://github.com/elastic/logstash	2010	464
Thanos	https://thanos.io/ https://github.com/thanos-io/thanos	2017	359
Prometheus AlertManager	https://prometheus.io/docs/alerting/latest/alertmanager/ https://github.com/prometheus/alertmanager	2013	222
Jaeger	https://www.jaegertracing.io/ https://github.com/jaegertracing	2016	216
Metricbeat	https://www.elastic.co/beats/metricbeat https://github.com/elastic/beats/tree/master/metricbeat	2014	100
Weave Scope	https://www.weave.works/oss/scope/ https://github.com/weaveworks/scope	2015	97
Graphite	https://graphiteapp.org/ https://github.com/graphite-project	2008	23
Zabbix	https://www.zabbix.com/ https://github.com/zabbix/zabbix	2011	7
Cortex	https://cortexmetrics.io/ https://github.com/cortexproject	2016	5

Table 1 Open-source projects relevant to cost monitoring, the year when the project was started, and the number of contributors in June 2021

3.1 Monitoring and logging

3.1.1 Prometheus

Prometheus is an open-source service monitoring system used to monitor events, trigger aggregation operations on collected data, evaluate monitoring rules, and send warning / alert notifications. It is easy to use and supports a large number of possible integrations. It has become one of the most widely used monitoring tools. It has incorporated a simple and straightforward graphical user interface, allowing graph creation, editing configuration, and rule

creation. Prometheus is based on the Go programming language, which guarantees high search performance. Prometheus is one of the so-called time series monitoring systems. This means that all acquired data are time-stamped. Figure 7 shows the architecture of Prometheus. [51, 52]

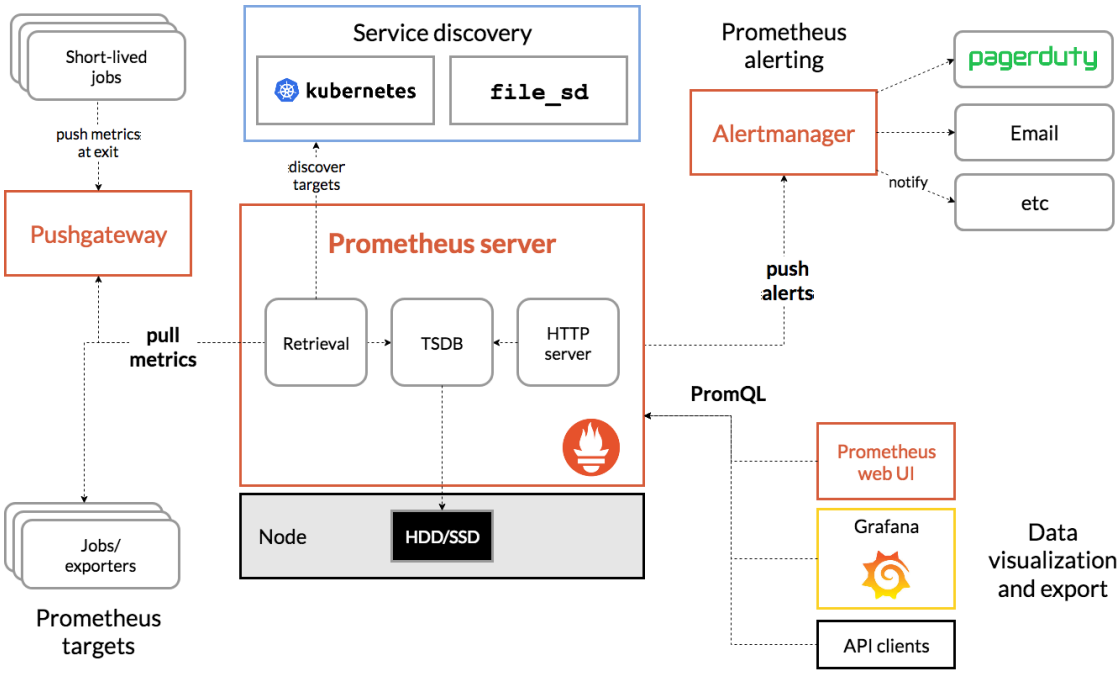

Figure 7 Prometheus architecture [53]

The following section describes the architecture of the Prometheus monitoring tool in more detail.

- The Prometheus server scrapes and stores time series data from monitored targets. Prometheus provides its own local database for time storage organised data and an interface that can be used for integration with other databases. TSDB stands for ‘time series database’.
- The Pushgateway supports short-lived jobs that may not exist long enough to be scraped. By default, Prometheus retrieves data using the ‘pull’ method, meaning that it is configured so that it knows when to scrape the data. However, some tasks or processes may be so short that they are not caught at runtime. For these cases, it is possible to use the so-called ‘push’ method, where the running process provides data to the component Pushgateway, which will then expose the data for Prometheus.

- Exporters are third-party tools designed to obtain information about the running application and then adapt its format for further processing. Most exporters are available for download for use as Docker containers. Special-purpose exporters are used when it is not feasible to instrument a given system with Prometheus metrics directly. Service Discovery is a tool used to detect components that should be monitored automatically. [54]
- Alertmanager is an intermediary between the monitoring system and the user. Alertmanager is used for the processing and subsequent management of alert notifications. Notifications are then distributed according to the configuration. Some supported communication channels are e-mail, HipChat, PagerDuty, Slack, and Webhook. [55, 56]

Prometheus is available on Docker Hub. It uses the PromQL query language. Exporters have been made available by numerous applications. The support community is large, and guidance is up-to-date and readily available. It is a popular tool that is undergoing rapid development. However, Prometheus does not support storage and subsequent analysis of logs; neither it supports collection and work with non-numerical data. In itself, it is not suitable for long-term data storage. A significant strength of Prometheus is the out-of-the-box possibility for federation when a Prometheus server scrapes metrics from another Prometheus server. This feature allows for the hierarchical collection of data necessary for multi-region and multi-cloud environments. [57]

An example implementation is the monitoring of service meshes on separate Kubernetes clusters, each of which has its own Prometheus instance that is then linked to a global Prometheus instance. [58]

3.1.2 Graphite

Graphite stores numeric time series data and renders graphs of this data. It is programmed in Python 2 (the version of Python available in 2021 is 3.9), which does not require expensive and powerful hardware to start and operate. Graphite does not actively collect data, like Prometheus does, but instead passively waits for data provided by an external tool. The monitoring application must be configured to send data directly to Graphite. Graphite is merely a

database with query language and graphing features. Graphite has value when storing cluster data in the long term. It is used together with StatsD, a network daemon that listens for statistics. [59, 60, 61, 62, 63, 64]

The architecture of the Graphite monitoring tool is demonstrated in Figure 8.

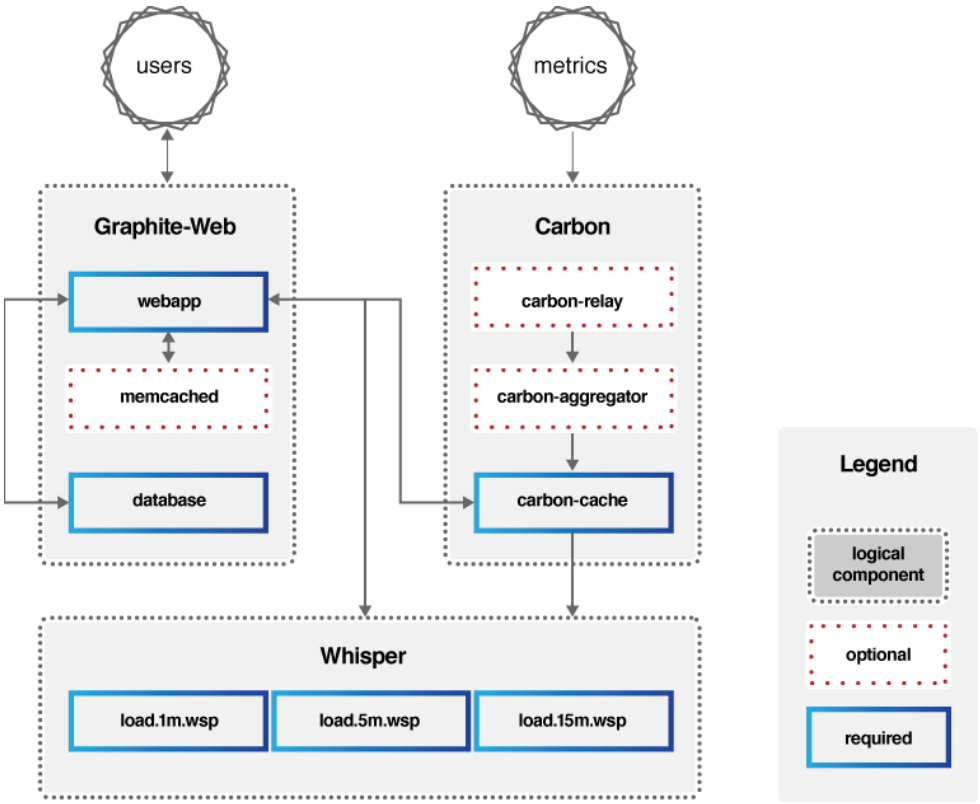

Figure 8 Graphite architecture [63]

Figure 9 shows the Graphite monitoring tool and its division into three main parts that handle data receipt, processing, and recording.

- Carbon is a service used primarily for receiving data and preparing data for storage. An optional carbon-delay component can write incoming data to two or more repositories as duplicates and sort incoming data.
- Whisper is a database based on the principle of the round-robin database. It provides fast and reliable data storage.
- Graphite-web is a graphical user interface for retrieving data from a Whisper database and display of the data.

According to its documentation, Graphite scales horizontally but does not provide functionality for federation, unlike Prometheus. [59, 63]

3.1.3 Thanos

Thanos and Cortex are Prometheus setups aiming to address some of the needs that arise from monitoring real-life production applications. Like Prometheus, Thanos and Cortex are also Cloud Native Computing Foundation (CNCF) projects, but they are in the so-called incubating stage. CNCF projects might be considered the most advanced open-source projects in cloud development. They create the basis for many commercial solutions. Therefore, the fact that setups like Thanos and Cortex are in the incubation stage may well indicate what is now possible to achieve in the cloud.

Thanos aims to overcome the short-term data storage of Prometheus. Thanos allows the storage of historical data in such a way that metrics can be queried globally across all Prometheus servers. It allows the storage of Prometheus time series data in object storage. Thanos can be added on top of existing Prometheus deployments and aims to work in situations where cross-cluster federation is used. Further, Thanos can aggregate data from Prometheus replicas. This is Thanos's solution to data gaps created when a Prometheus server becomes unavailable and data cannot be pulled. Another Prometheus instance is running in parallel, and Thanos claims to be able to aggregate the data without duplicates or errors. [65, 66, 67]

3.1.4 Cortex

As described in the previous chapter, Cortex is another CNCF incubation project. It aims to provide multi-tenant Prometheus as a service with authentication and authorisation functionality. This means that Cortex collects data across multiple machines in one cluster as illustrated in Figure 9. Furthermore, Cortex aggregates data centrally and uses a push-based method. This makes data available even when an edge location becomes unreachable.

This is an advantage compared to Thanos, which relies on pulling data and has nothing to pull if a Prometheus server becomes unavailable unless there is an unaffected duplicate Prometheus server. [68, 69, 70, 71]

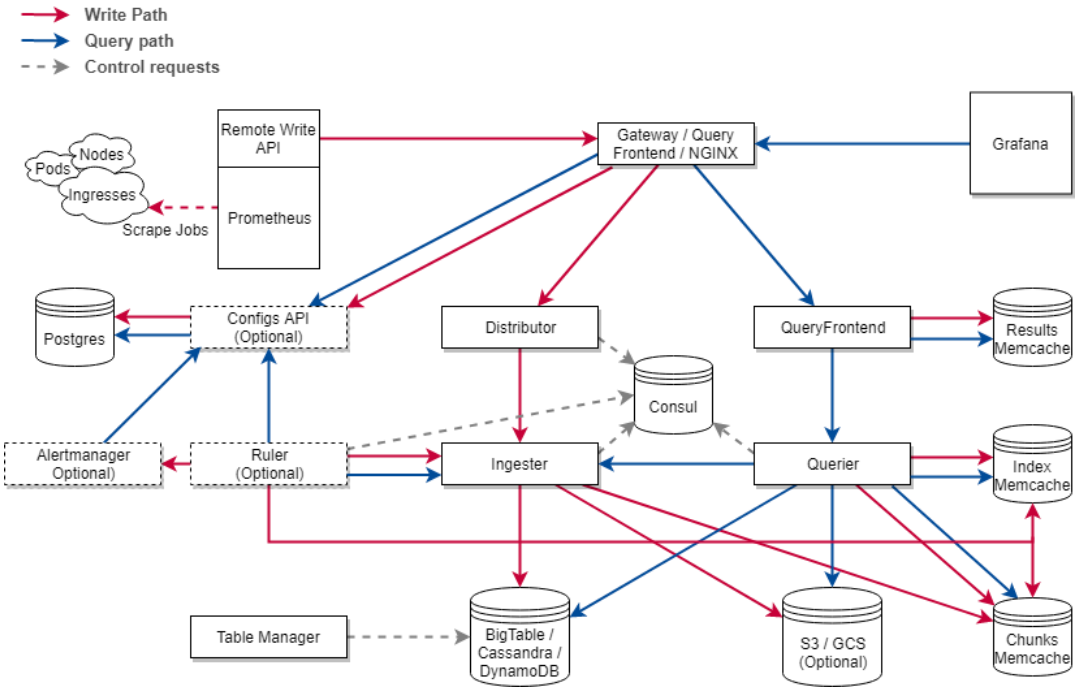

Figure 9 Cortex architecture [72]

3.1.5 Zabbix

Zabbix is a monitoring tool that promises metric collection, anomaly detection with alerts, visualisation, easy deployment and distributed monitoring. It is maintained by the GitHub community, counting eight contributors (June 2021). Its fork, Zabbix Docker Monitoring, counts 13 contributors and had its last commit in February 2021. As such, Zabbix serves as an example of an open-source project that has not received enough traction and seems to be disappearing under the shadow of larger and more successful communities. However, as the forces of open-source change very quickly, it may be an advantage to know all alternatives. [73, 74]

Zabbix architecture

The Zabbix monitoring system can be deployed and operated by installation on a server.

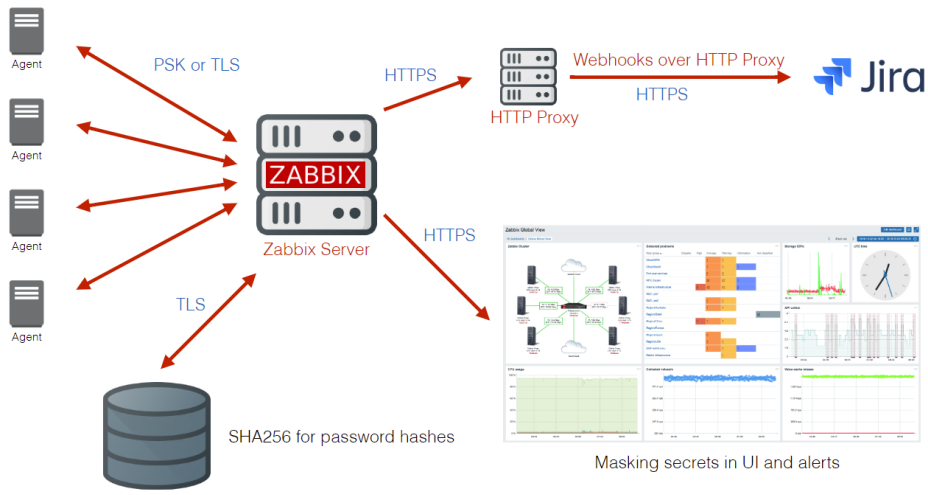

Figure 10 Zabbix architecture [75]

As shown in Figure 10, the Zabbix server is the central component responsible for managing the entire monitoring process. The server is responsible for processing incoming data, defining monitoring tasks, defining alert triggers, and sending notifications. Zabbix supports several types of databases: MySQL, Oracle, PostgreSQL, TimescaleDB, IBM DB2, SQLite. [75, 76]

A Zabbix proxy collects incoming data sent from one or more monitored devices. The data are stored in the cache and only then transferred to the Zabbix server. The Zabbix agent can be configured to work in two modes:

- Passive – Passive mode works on the request/response principle. The agent responds to a data request sent from a Zabbix server.
- Active – In the case of an active configuration at predetermined intervals, the agent periodically sends information about monitored items.

Zabbix was originally developed for monitoring servers. Zabbix, unlike Prometheus, can provide out-of-the-box functionality for storing and analysing

text values and trigger alerts with advanced functionality as alert escalation. According to Metricfire, Zabbix is the best choice in the following scenarios:

- There is a need for open-source software with a C back end and PHP front end.
- The data are stored in MySQL, MariaDB, PostgreSQL, SQLite, Oracle, or IBM DB2.
- There are less than 1000 devices.
- There is a need for monitoring but not necessarily for outstanding visualisation. [77]

Zabbix claims that it can monitor multi-tenant environments and can be used together with Grafana for visualisation. It is possible to add custom agent checks, but there is nothing equivalent to the portfolio of Prometheus exporters. Altogether, no evidence could be found that Zabbix is suitable for anything beyond network monitoring. [78]

3.1.6 Telegraf

Figure 11 highlights how Telegraf monitoring agent is used to collect, process and aggregate time series data from a variety of sources. Telegraf collects metrics from multiple devices, systems, or IoT sensors. It is written in the GO language and is plug-in based. This allows the collection and output of data without further modification. [79, 80]

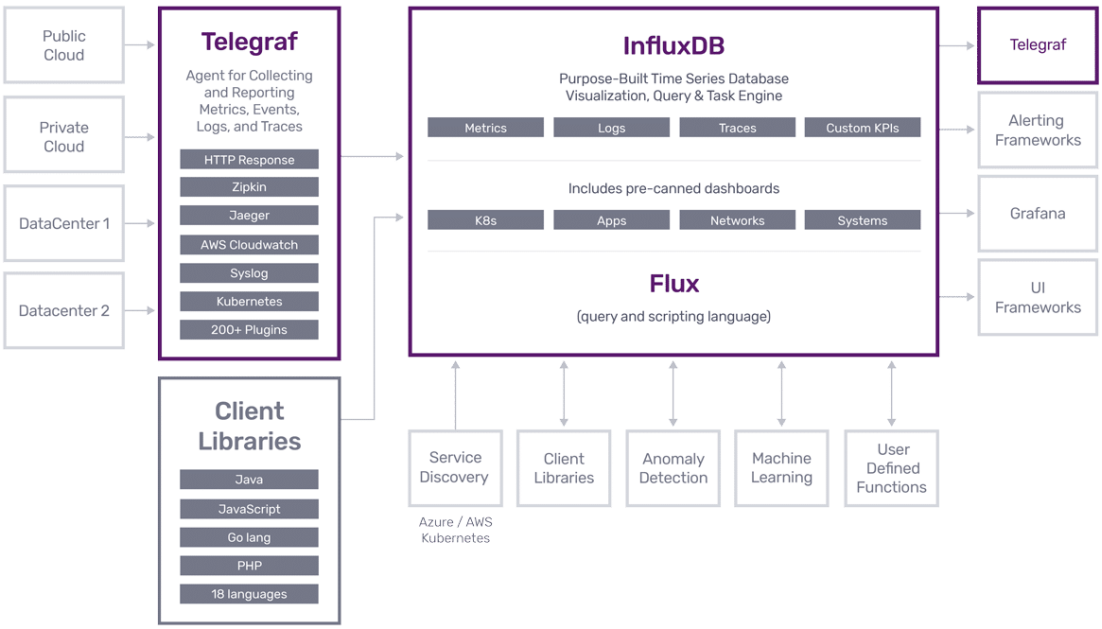

Figure 11 Telegraf architecture [79]

3.1.7 Elasticsearch

Elasticsearch is an analytical, searchable, NoSQL database programmed in Java. It is currently one of the most popular available database systems. Its main domains include searching, analysing, and processing large amounts of data in a short time. The search is based on the open-source technology Apache Lucene. [81, 82] Since Elasticsearch is used to collect logs and it is not certain whether this feature is useful for cost-related metering, there will be no further analysis of this tool.

Elasticsearch was relicensed in 2021 from an open-source license [38, 39, 43]. More on the topic of open source is briefly discussed in Chapter 2.11, including references to other projects that abandoned the open-source strategy and discussions of their reasoning.

3.1.8 Logstash

A tool designed to collect data from several sources at once. Logstash supports a large number of possible inputs, such as Beats. Data are then sent to the configured source, most often to the Elasticsearch database. Other available targets for data collection include the following:

- Graphite
- JDBC (Java Database Connectivity)
- Unix

Beats – Logstash extension: These are lightweight agents used to collect data and send data. It is possible to use several types of collection agents from different parts of the infrastructure:

- Filebeat retrieves data from files. It can be used effectively for collecting application logs.
- Metricbeat is a lightweight metric collector. It collects metrics from the system and services on the server. After collecting and sending metrics to Elasticsearch or Logstash, Kibana can be used for visualisation.
- Packbeat is a lightweight agent used to analyse network protocols.
- Winlogbeat is an agent designed for retrieving system logs from the Windows operating log.
- Auditbeat uses the same principle as Winlogbeat, except that it is designed for Unix operating systems.
- Heartbeat is an agent used to monitor other components using repeated inquiries. Queries can be performed using ICMP (Internet Control Message Protocol), TCP (Transmission Control Protocol), and HTTP. [83, 84]

3.2 Visualisation

Tool	Website GitHub	Year started	Contributors
Grafana	https://grafana.com/ https://github.com/grafana/	2013	1500
Kibana	https://www.elastic.co/kibana https://github.com/elastic/kibana	2013	700
Chronograph	https://www.influxdata.com/time-series-platform/chronograf/ https://github.com/influxdata/chronograf	2016	81
Weave Scope	https://www.weave.works/oss/scope/ https://github.com/weaveworks/scope	2015	97

3.2.1 Grafana

Grafana is an open-source tool used to visualise and create dashboards over already acquired data. Grafana does not store the collected data but only draws data from the configured source, over which it is then possible to perform visualisations, analysis, and alerts. Grafana's significant advantage is database independence. The graphic design is at a very high level, as can be seen on Figure 12. It is easy to use, and intuitive. Another advantage is the possibility of exporting already-created dashboards and visualisations and their subsequent sharing between projects, provided they use the same types of metrics. However, the platform does not allow full-text data querying and cannot, therefore, fully replace Kibana. Grafana Loki is a suitable tool for logging. Instead of indexing logs, it uses labels that make it suitable for the Kubernetes environment. [85, 86]

Figure 12 A showcase of dashboards in Grafana [85]

3.2.2 Kibana

Kibana is an open-source log analysis platform that allows the exploring, visualising, and building of dashboards on top of the log data stored in Elasticsearch clusters. It allows the creation of new views, tables, maps, and graphs over stored data as shown in Figure 13. Data can be plotted in real time. Unlike Grafana, Kibana Is used primarily for analysing log messages (full-text data querying) and offers functionality for troubleshooting, forensics, development, and security. Similar to Prometheus, Kibana has extensive alert management. [87, 88, 89]

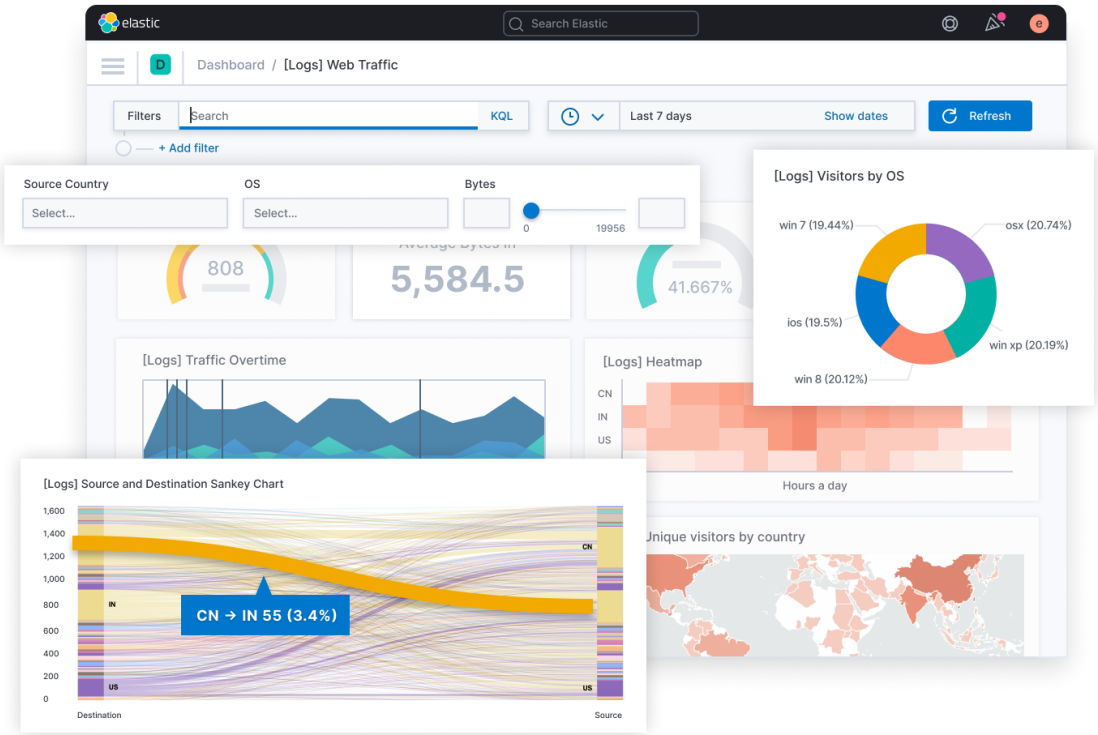

Figure 13 A showcase of dashboards in Kibana [87]

3.2.3 Chronograph

Chronograph is a real-time visualisation tool for data stored in the InfluxDB database. Chronograph is used to create new views, as shown in Figure 14, on data and start-up rules. The application is programmed in the GO programming language and provides fast writing, availability, and data reading. [90, 91]

Figure 14 A showcase of dashboards in Chronograph [92]

3.2.4 Weave Scope

A visualisation and monitoring tool for Kubernetes and Docker, Weave Scope offers a top-down view of an entire infrastructure, as shown in Figure 15. Developed by Weaveworks, Weave Scope generates a map of processes, containers, and hosts in a Kubernetes cluster. Its graphical user interface allows the management and running of diagnostic commands on containers. [93, 94, 95]

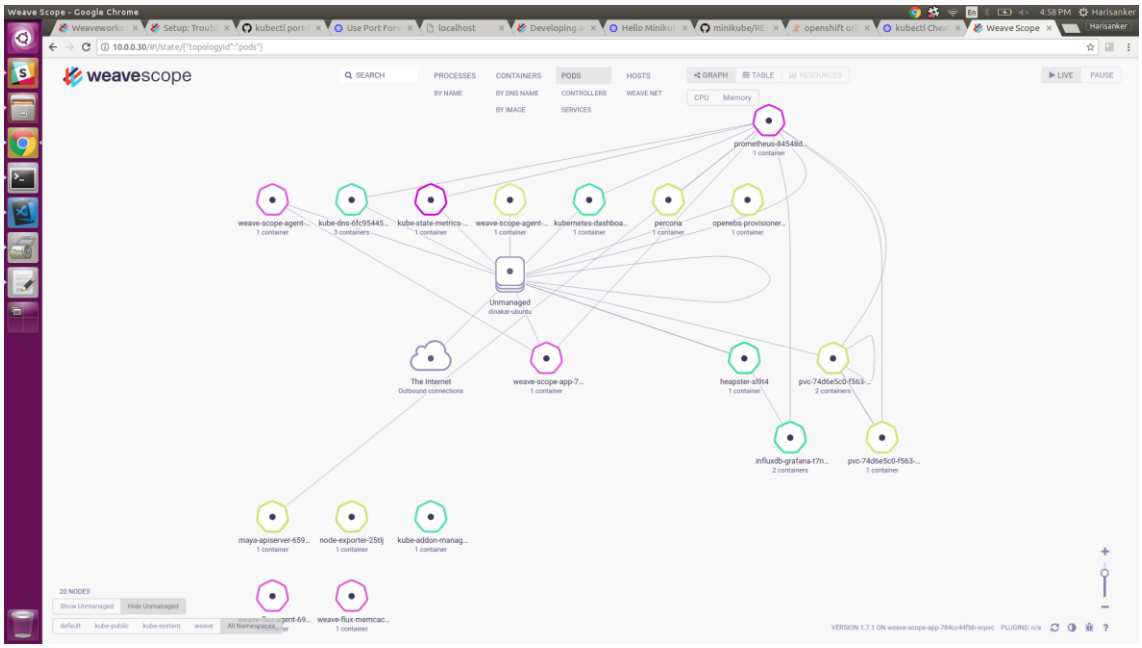

Figure 15 A showcase of dashboards in Weave Scope [96]

3.3 Stacks

Many open-source tools were built with the primary intention of being used with specific other tools. Such tools have evolved together, and as a result, it is considerably easier to begin using a complete stack rather than using random tools together. Complementing functionality and easy deployment are advantages of using stacks. The most prominent stack on the open-source market is the TICK stack.

3.3.1 TICK – Telegraf, Chronograph, InfluxDB, Kapacitor

The TICK Stack provides storage and visualisation of timed data and subsequent notifications. Its high level architecture is on figure 16. The main component of the architecture is Telegraf, the TICK stack uses the push method, which is suitable for IoT monitoring and real-time analysis. A Telegraf monitoring agent is used to collect time series data from a variety of sources. InfluxDB stores time series data. Chronograph visualises time series data from the InfluxDB database.

InfluxDB, like Telegraf, is written in the GO language and is a database designed for storing chronological data. Some functionality is reserved for the InfluxDB Enterprise.

Kapacitor is an InfluxDB data-processing tool implemented in the GO programming language. Kapacitor triggers warnings, runs the extract, transform and load processes (ETL), provides anomaly detection, and responds to events. It can be used for real-time data processing and analysis before data are saved to the InfluxDB database. Alternatively, it can be used for system integration to detect anomalies. Notifications can be sent using the following communication channels: HipChat, Alert, Sensu, and PagerDuty. [97]

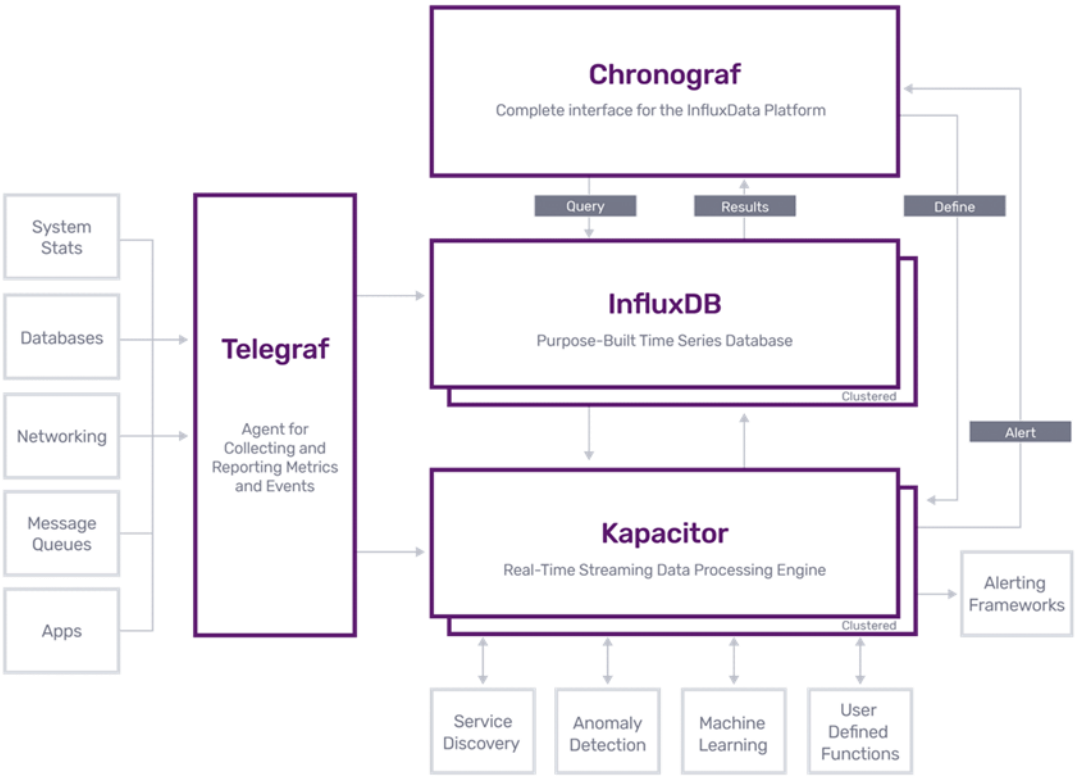

Figure 16 TICK stack architecture [97]

3.3.2 ELK – Elasticsearch, Logstash, Kibana

ELK is a monitoring solution for data processing and analysis, mainly for application logs. The combination of components is used for storage, processing, retrieval, and visualisation. Individual components are illustrated in Figure 17. Data are collected through Beats, Elastic Agents, or Logstash. All mentioned components are open source and are available on the Docker Hub. The ELK stack offers over 200 integrations with other tools and is offered by AWS and Google Cloud as a managed service and integrates with Azure. [98, 99, 100, 101]

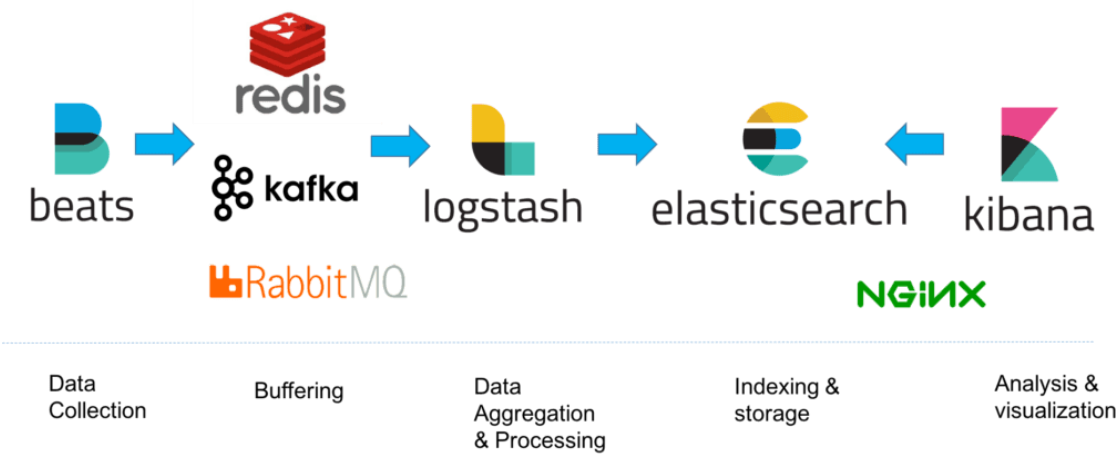

Figure 17 Example of an ELK stack toolchain [102]

An example variation of the ELK stack is the Elasticsearch, Metricbeat, Grafana (EMG) stack. Metricbeat is the Beat with which metrics are collected. Data are visualised with Grafana instead of Kibana. [103]

3.3.3 EFK – Elasticsearch, Fluentd, Kibana

The EFK stack replaces Logstash with Fluentd. A detailed comparison of both is discussed here [104]. The main difference between the ELK and EFK stacks is that EFK does not seem to be maintained by any commercial party. Further, EFK is currently an implementation meant to work mainly with Kubernetes

rather than a complex environment that readily collects data from multiple sources. Elastic actively maintains the ELK stack. The EFK stack is centred around Fluentd, which offers multiple integrations for data collection. Still, it does not look as though Fluentd takes ownership of the maintenance of the stack as a complete logging solution. Users must thus rely on separate helm charts or solutions prepared by independent parties. [105, 106, 107] Figure 18 illustrates collecting of data from Docker containers, and their further processing in Elasticsearch, Fluentd and Kibana.

Figure 18 EFK stack [108]

3.4 Commercial solutions

There are many commercial cloud monitoring tools and platforms available. These include the following tools, among others:

- Splunk [109]
- AppDynamics [110]
- Dynatrace [111]
- Instana [112]
- Sensu [113]
- Puppet [114]
- Azure Monitor [115]

- CloudWatch [116]
- Cloud Monitoring [117]

The list below is provides an overview of the commercial solutions that offer out-of-the-box capability for cost-related metering:

- CloudZero [118]
- Harness Cloud Cost Management [119, 120]
- Apptio [121]
- CloudCheckr [122]
- Cloudhealth [123]
- Computesoftware [124]
- Parkmycloud [125]
- AWS Cost Explorer and AWS Budget [126]
- GCP Billing [127]
- Azure Cost Management [128]
- Kubecost [129]
- Spot.io [130]
- Turbonomic [131]
- Yotascale [132]

It is beyond the scope of this thesis to analyse the commercial offerings in detail. However, their overall advantages will be briefly discussed in the next chapter.

4 Problem analysis and proposed solution

The following subchapters describe the key cost contributors in SaaS, explain in detail the kind of information a company might need to gather, and presents viable approaches for gathering and processing the necessary data.

It should be mentioned that the further proposed solutions are meant to create an integrated part of a SaaS platform. In this case, the SaaS platform is not meant to be a cloud solution that can be adapted to different needs but rather a central point of access, control, monitoring, and management of services. An example can be seen in Figure 19, which shows OKD, the community distribution of Kubernetes that powers Red Hat OpenShif. OKD does not include cost metering but illustrates many of the other possible metering needs of Kubernetes-based business. Other parts of such a platform beyond cost metering depend on the service provider but would likely at least include functionality as a service catalogue, non-cost related service monitoring, access management, and so forth.

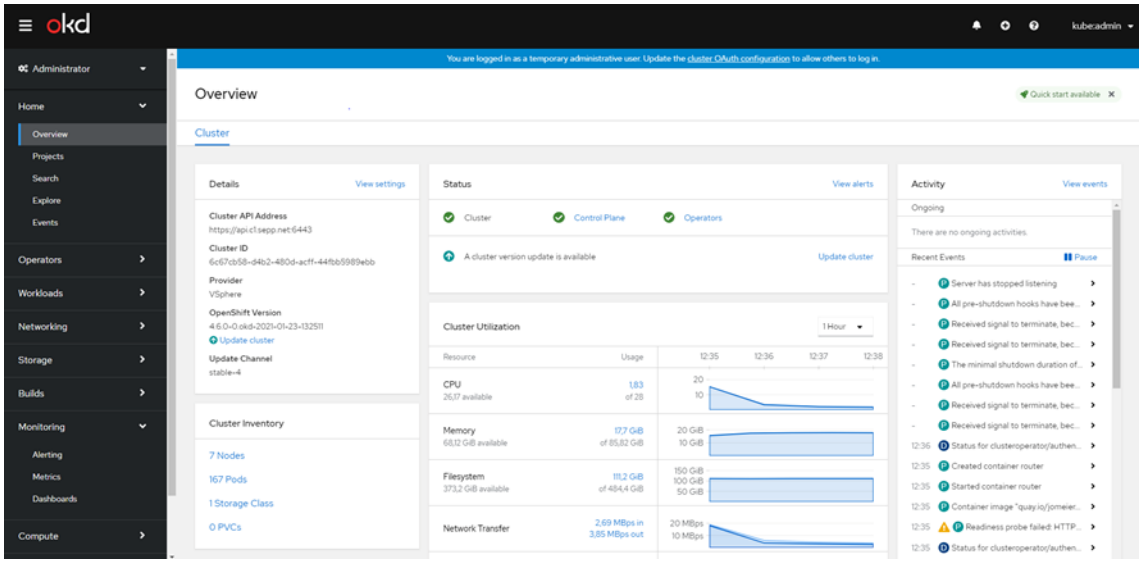

Figure 19 OKD example [133]

When monitoring costs, part of the data used to generate the final reports is common with other monitoring, for example with workload monitoring. Cost-

related data collection and reporting may take different approaches when it comes to integration into the platform. On a high level, the following approaches illustrated in Figures 20 and 21 are possible:

a) The base tools, data collection, and data processing can be the same for all other monitoring needs. The only unique feature for cost analysis is the final dashboard as an output of the collected data.

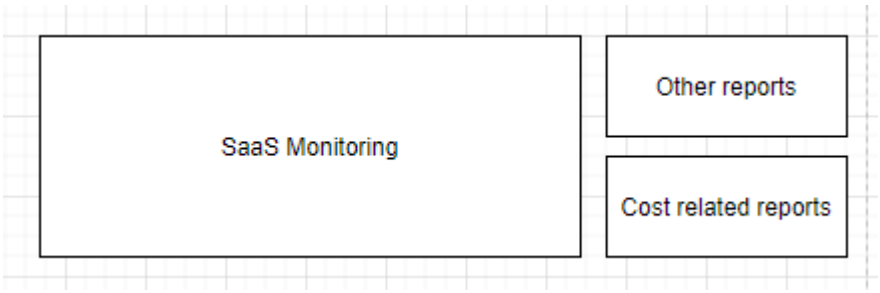

Figure 20 SaaS monitoring common for all reports

b) Cost analysis can be a self-standing module that is not connected to other types of monitoring. However, this approach would lead to much higher costs for monitoring as parts of the collected data are the same for all monitoring.

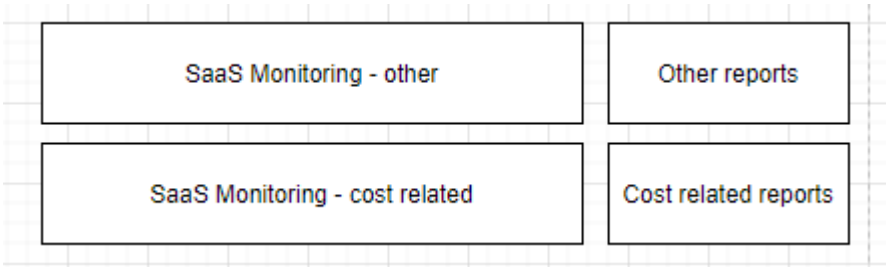

Figure 21 SaaS cost-related monitoring separate from other monitoring

4.1 Key cost contributors

There are three main cost contributors in the hybrid cloud: public cloud costs, private cloud costs, and third party solution costs.

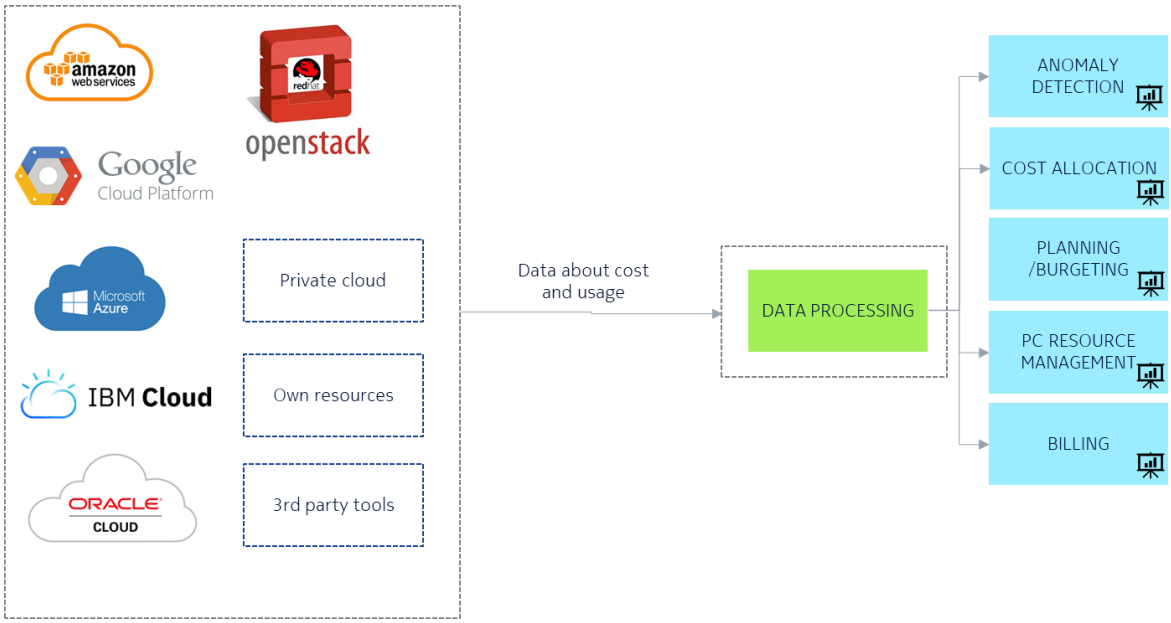

Figure 22 Key cost contributors and key capabilities

On the left side of Figure 22 above are listed the main cost contributors of SaaS. The main costs are those paid for the public cloud, a private cloud, and possibly third-party costs for any application.

This project does not consider the following costs:

- costs for research and development
- costs of human resources
- costs for the transfer to the public cloud, whether a price charged by the public cloud provider or the time costs of employees that conduct the transfer

The objective of this analysis is to look at the costs of a service that is ready to be sold to customers and to allocate these costs to customers, the public cloud service, functionality, or a team within the developing company. This may include both fixed and variable costs as:

- Storage fee.
- Transaction costs (costs for access to one's own data). For most public providers, these costs are in the range of cents, but they can rise to higher numbers for many applications and customers.

- Data transfer charges, incurred mostly through outbound data transfer sand transfers between regions and zones.
- Costs for the transfer of a solution from one public cloud to another.

It is cheap or free to start with or transfer data to the public cloud. As a business grows and data are more frequently accessed or transferred, public cloud costs rise beyond expectations. The added value is that if a service is not proven viable, it can be pulled away from a public cloud. One still must ensure that there are no unused assets left behind that would generate costs.

The public cloud cost structure is available online, and all public providers offer cost calculators. However, for cost calculators to provide reliable information, one must accurately know the consumption details of all the necessary individual tools. There is no practical way of accurately estimating these details for complex applications with high volumes. Furthermore, in the public cloud, as elsewhere, prices are negotiable for large customers. Current prices can be seen online for major public cloud providers:

- Amazon Web Services Pricing Calculator [134]
- Google Cloud Platform Pricing Calculator [135]
- Microsoft Azure Pricing Calculator [136]
- IBM Cloud Cost Estimator [137]
- Alibaba Cloud Pricing [138]

Private cloud costs include the price of hardware, software, and human resources, assuming that servers can be set in-house. Whether the private or the public cloud is cheaper is influenced by many variables. Perhaps the most informative analysis was made by VMware [139].

4.2 Required capabilities

There are several different customers and use cases for cost-related data in a business.

4.2.1 Anomaly detection

There is a strong need for companies to quickly recognise when anything in a service they provide is becoming out of the ordinary. This need also applies to cost management. Anomaly detection, as shown on example in Figure 23, searches for unexpected spikes in costs and sends out warnings. To significantly increase service reliability, anomaly detection is generally expected to occur in real time. Further, when managing a whole portfolio of services across several clouds and multiple regions, alerts must be set so that operators are not overwhelmed by unnecessary or false-positive warnings. This would lower the attention given to such messages very quickly. On the other hand, a lack of attention when setting alerts could easily lead to false negatives, that is, to completely missing important alerts. Artificial intelligence and machine learning are therefore used to adapt thresholds in the ever-evolving system.

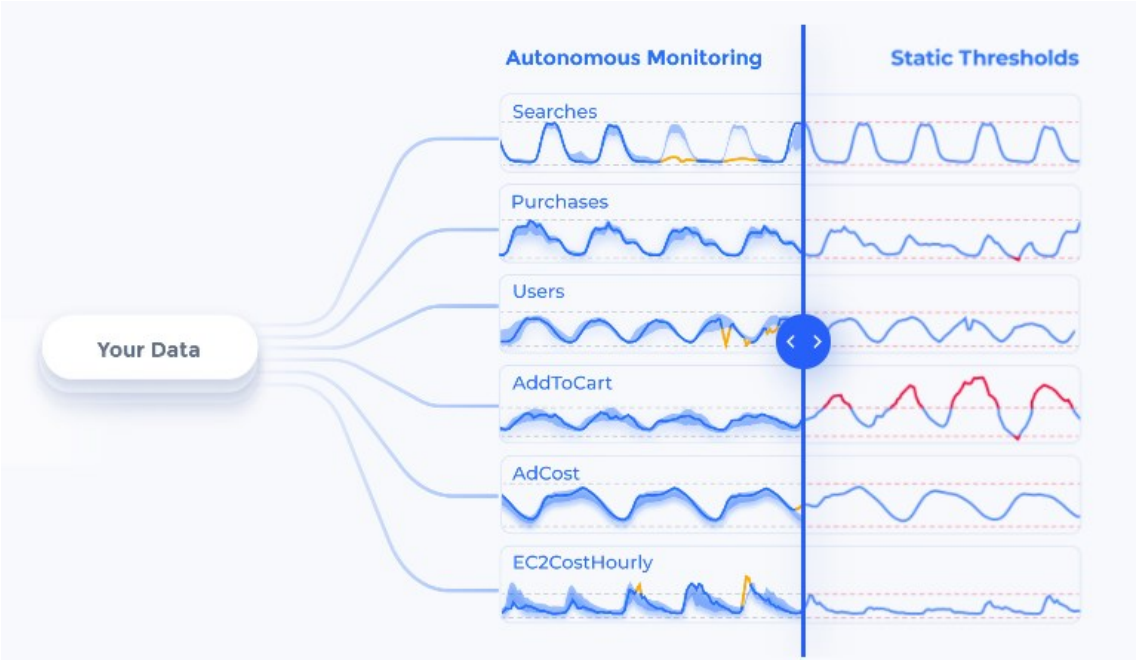

Figure 23 Anomaly detection by Anodot [140]

4.2.2 Cost allocation

Cost allocation is a tool for product managers and developers. In this context, it refers mainly to the allocation of costs to products and functionality, as illustrated in Figure 24. When developing a service that is expected to compete in price, it is essential to optimise costs. Before data can be optimised, it must be analysed with a sufficient level of detail. The data analysis must provide granularity at the levels of service, microservice, customer, region, time, and public cloud product. Then it is possible to find the places where spending occurs. Analysing data with such granularity makes it possible to see the costs of a new version of software before it is deployed into production (if possible and practical), but mainly allows one to quickly see the costs of canary deployments. Canary deployment refers to the deployment of a new version of a service into a small percentage of pods, for the purpose of obtaining production data on a sample of users.

To summarise, costs that have sufficient granularity will enable allocation that enables:

- find the biggest cost contributors,
- make a connection between the microservice and the public cloud product contributes to costs,
- provide a base of data for anomaly detection, and
- verify the costs of new service versions.

To achieve such an analysis, it is necessary to track costs and usage and combine them. The relationship between usage and cost data can be tracked with tags. In Kubernetes, tags are necessary for container orchestration allowing interaction between pods and containers. To obtain reliable and consistent data, it is necessary to use tagging consistently across all services and customers.

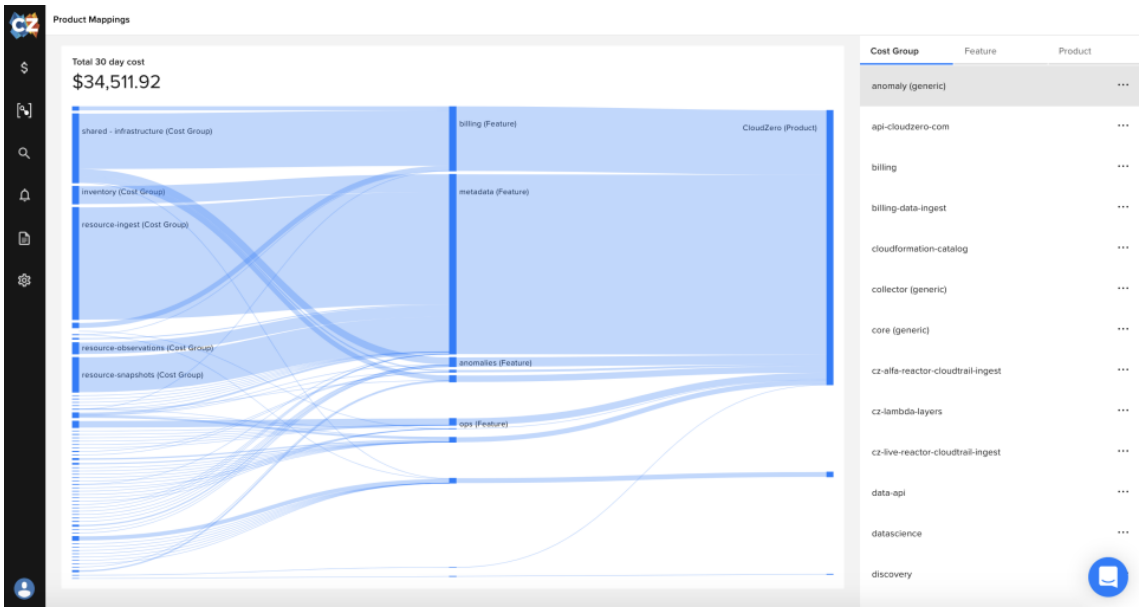

Figure 24 Example of allocation of costs per product feature [141]

4.2.3 Planning and budgeting

Another customer for SaaS cost information is management and top management, who need past data to forecast future expenses and earnings. An example of information presented on such level is in Figure 25. Again, the data should be granular to the level of product, customer, region, and cost centre. The data do not need to be real-time, but there should be functionality for generating forecasts and connecting to the corporate budget structure.

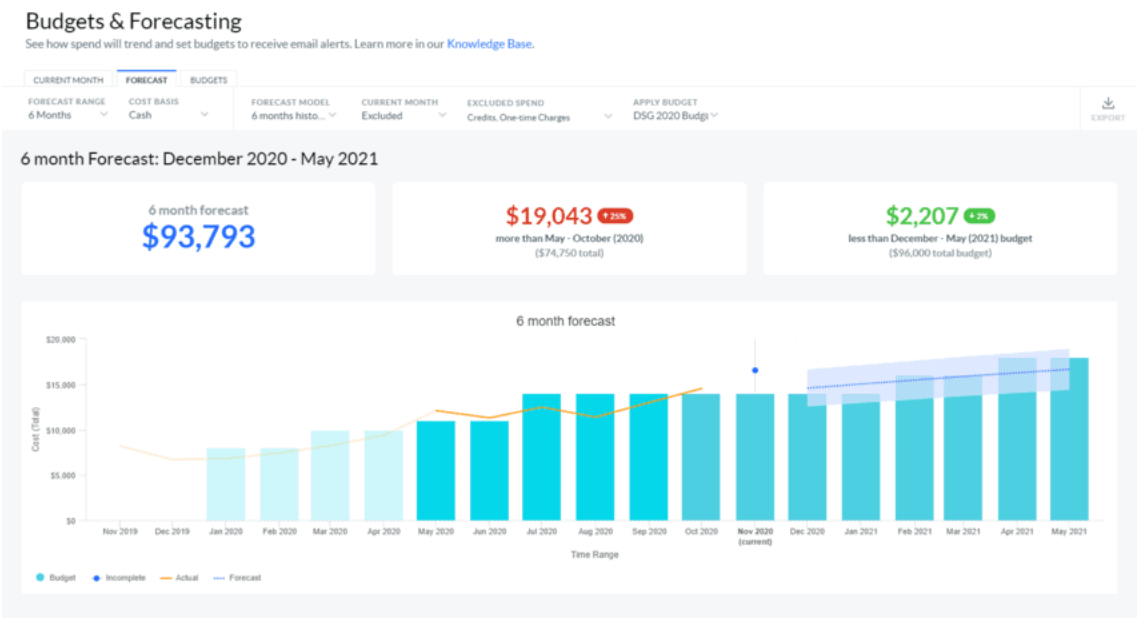

Figure 25 Budgets and forecasts example by Apptio [142]

4.2.4 Resource management

The previous subchapter discussed that product managers should know the costs their products generate, and top management needs the same information on a higher level to oversee all services. Beyond this, team managers need to know how much their team spends on the cloud, and deployment managers need to consider deployment from a cost-saving perspective. Therefore, data should be granular at the levels of teams, services, public cloud products, and time. An example of such implementation is shown on figure 26. The objective is to ensure that cloud resources beyond those used for running production code are used efficiently. Many other needs in a company require the public cloud beyond just running production code, including learning, research and development, and testing.

The goal is to make sure that resources are not being spent unnecessarily on the following costs:

- unused and non-responsive instances, that is, instances of public cloud services that have been requested but are no longer actively used
- unattached persistent volumes

- orphaned volume snapshots
- costs for testing and deployment
- unused static IP addresses
- underutilised reserved capacity
- any other spending that does not directly relate to the product

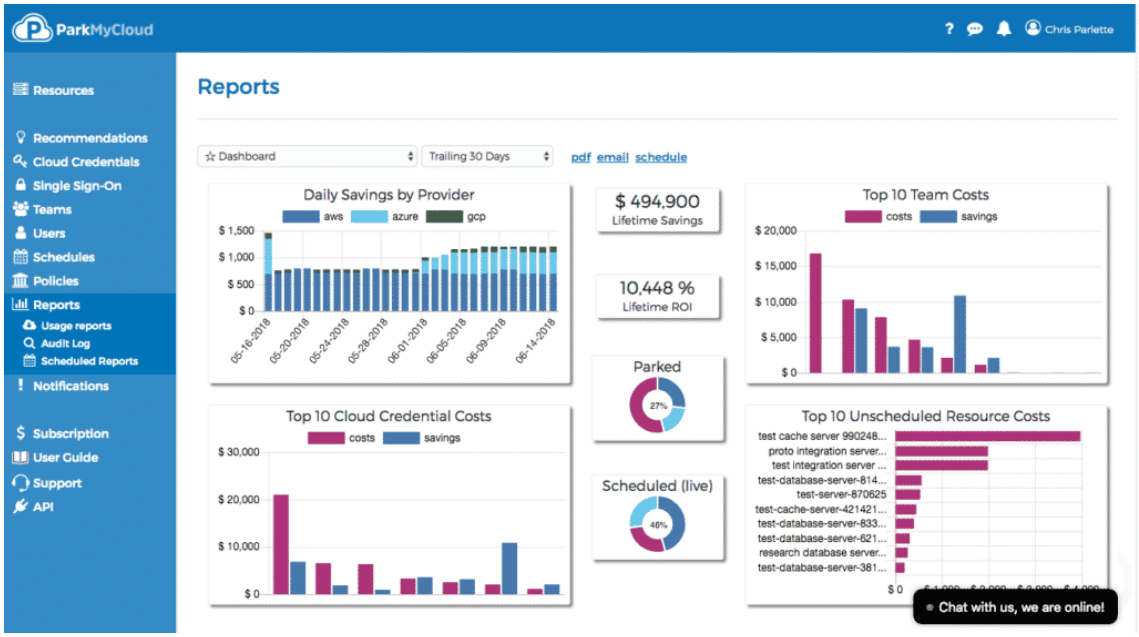

Figure 26 Cloud costs reports by ParkMyCloud [143]

4.2.5 Billing

Billing on its own may or may not be a consumer of cost-related information. The case of on-demand billing is the same as the cost analysis mentioned in Chapter 4.3.2; that is, it is a consumer of data about usage. In the case of billing, ‘costs’ refer to costs for the business customer, that is, the price customers pay for the service in the cloud. This chapter explores whether costs to the service provider relate to costs for the customer.

According to Iveroth et al. [10], the pricing model that a company chooses for its service has several parameters: scope, base, influence, formula, temporal rights, degree of discrimination (different prices for different customers), and dynamic pricing strategy. Therefore, management has many options for how to position

their pricing strategy. In addition, there can be volume discounts and on-demand vs reserved capacity pricing. According to the research, some of these options are more popular than others.

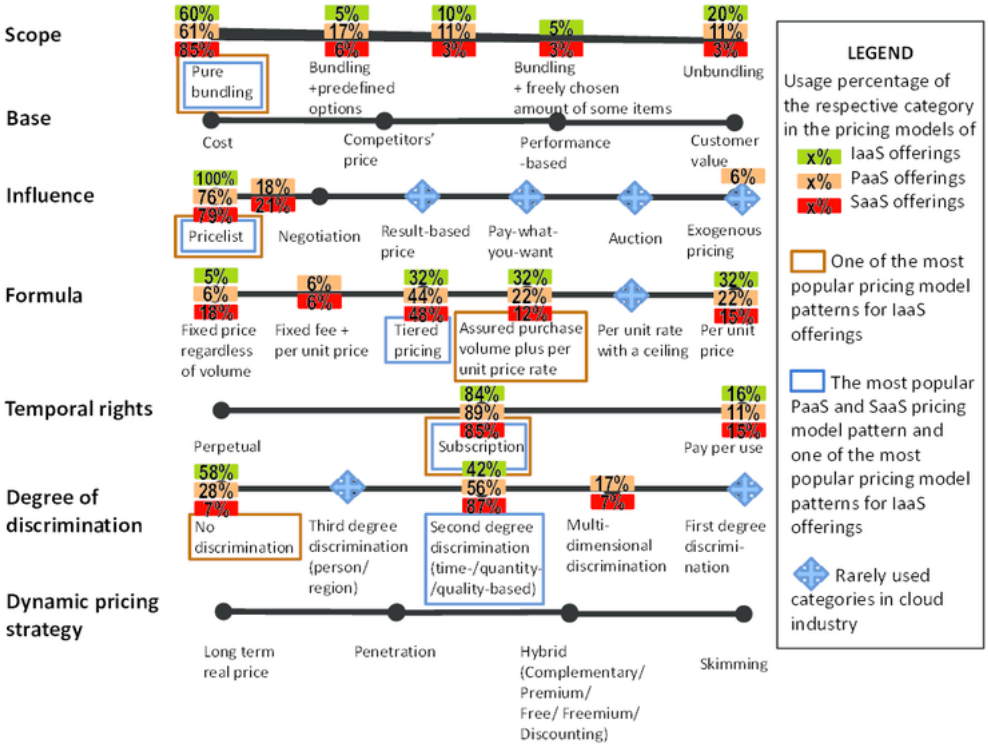

Figure 27 Current pricing models in the cloud industry [144]

As can be seen in Figure 27 above, the most popular pricing model for SaaS is tiered subscriptions, that is, a pricing model in which a customer pays a fixed monthly fee based on a pre-agreed volume that has been allocated to the customer. There are no separate reports needed for this kind of pricing, as all prices are likely to be set with data from planning and budgeting (see Chapter 4.3.3). However, there is about a 15% chance that the price will be pay-per-use. Charging based on actual usage means that customers pay nothing if they use nothing (and usually customers pay nothing up to a certain volume, the so-called 'free tier'). In practice, usage is expressed as a volume of selected metric(s) or a measurable added value delivered to the customer (e.g., savings). Therefore, it is not justified to link price directly to the provider's costs. The reason for this is that costs can change suddenly when a changed service is deployed. For

example, when a provider introduces new monitoring functionality, public cloud costs increase because of the extra data that need processing. Further, shifts in consumption of public cloud services may move the provider into a different pricing tier. When it comes to usage, the following types of metrics are common:

- Number of requests, items (hosts, cluster, etc.), instances, queries, and users
- Number of something per instance
- Amount of something per instance
- Type of request/item
- Average size of request/item
- Average duration of connection/instance
- Average connection rate
- Data amount per query/item
- Number of hours of connection days users access the product
- Percentage of users who use (some) functionality X days per month
- Lines of code (LOC) in one file

In addition, price is specified per product, functionality, region, and time.

4.3 Solutions

In Chapter 3, a portfolio of different potential tools was introduced. The goal is to find a suitable service, toolchain, or architecture that would enable metering in a cloud that provides the capabilities described in Chapter 4.2 and would work in multiple environments:

- multi-cloud
- hybrid-cloud
- multi-tenant
- multi-region
- Kubernetes-based.

There are three basic approaches when choosing tools for cost metering: a combination of open-source tools; a paid service; or a special case of the latter, a custom solution consisting of public cloud services.

Commercial solutions

Commercial solution providers have the advantage of know-how and experience. Their portfolios can range from including one required capability, as listed in the previous subchapter, up to including complete Technology Business Management (TBM) [145].

The reasons to consider commercial solutions, even when the primary focus is on open source, are as follows:

- The ability to start cost metering early. When a company is in an early stage of providing SaaS and employees are still learning basic functionality, cost management can easily fall out of focus.
- The possibility of learning from the service provider.
- Access to best practices

Public cloud solution

A special case of commercial solutions are services from public cloud providers. Public cloud cost analysis tools currently only allow the analysis of data from their own clouds; for instance, AWS Cost Explorer and AWS Budget, Google Cloud Billing, and Azure Cost Management only manage costs in AWS, Google Cloud, and Azure, respectively. The advantage is that basic functionality is available as a free part of other subscriptions. To combine cost metrics from different clouds, cloud providers only offer generic monitoring solutions. These solutions are not meant for cost analysis but for metering or data processing in general. Google Cloud provides a complete recipe for processing data in the cloud. Its schema can be seen in Figure 28, which lays out the full portfolio of tools that can be used for advanced data analytics. [146]

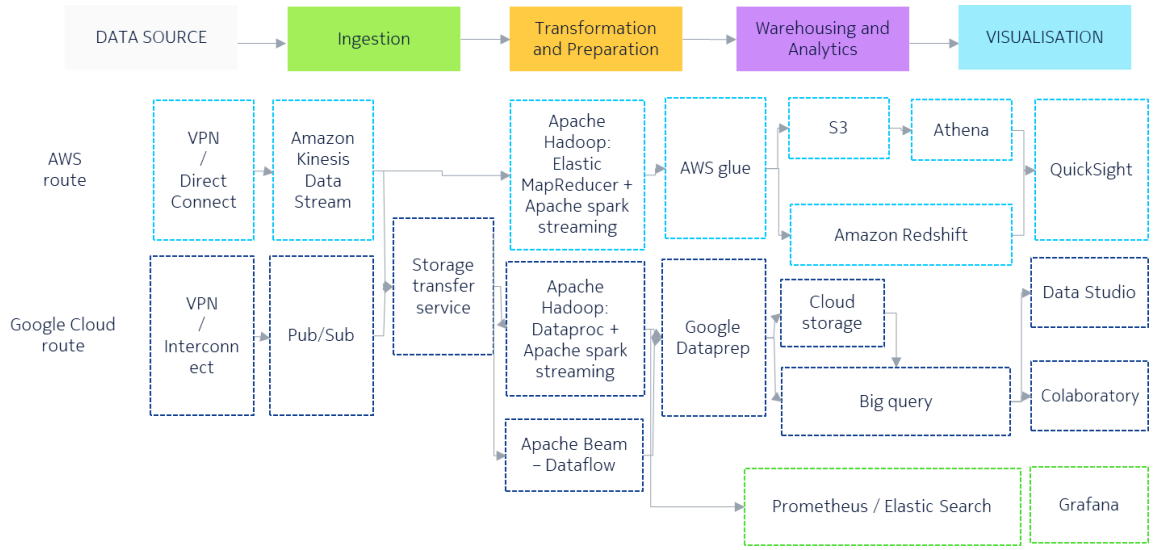

Figure 28 Public cloud data analysis toolchains [146, 147]

Open-source solution

Since Docker and Kubernetes are open-source tools. The challenges of committing to the open-source route were discussed in Chapter 2.12. However, one indisputable advantage is that open source is currently not just a pool of isolated tools but a whole ecosystem of interrelated solutions. However, a prerequisite for using open source is the willingness to contribute to the open source in cases of significant changes in ‘offerings’ or continuing developments in-house, in the case of relicensing.

After careful consideration of all the positive and negative aspects of the selected monitoring tools and their usability for this work, Prometheus appears to be the best monitoring tool. It is an open-source technology, so it is affordable and economically undemanding. It is easy to use and has high community support (contributions to future solutions), and many exporters are available. Combined with the visualisation tool Grafana, Prometheus provides a powerful tool that can process and display almost any kind of metrics, not only cost-related metrics. An additional tool that stands out is Kiali, which provides an easy-to-understand overview of microservices and therefore helps one to understand service structures at a glance. The Prometheus alert manager

stands out with its efficiency optimisation algorithms. As for metrics collection, data about Kubernetes go to Prometheus straight from the Kubernetes exporter, and data from services come from the services´ own exporters. These data contain usage information. Regarding data about costs, public clouds also readily integrate with Prometheus. For a private cloud, a combination of Prometheus and Fluentd could be used. It is more complex to collect data from edge devices. Here, tools such as CollectD and RabbitMQ are useful. Other tools that are part of the deployment of microservices, such as gRPC or Istio, also readily integrate with Prometheus. Furthermore, Prometheus can be deployed using the same methods as any other service, for example, with Terraform (open-source infrastructure as code software tool) and Helm (Helm Charts define, install, and upgrade Kubernetes applications). Figure 29 shows the toolchain for gathering data from public and private clouds up to visualisation, using the tools described above.

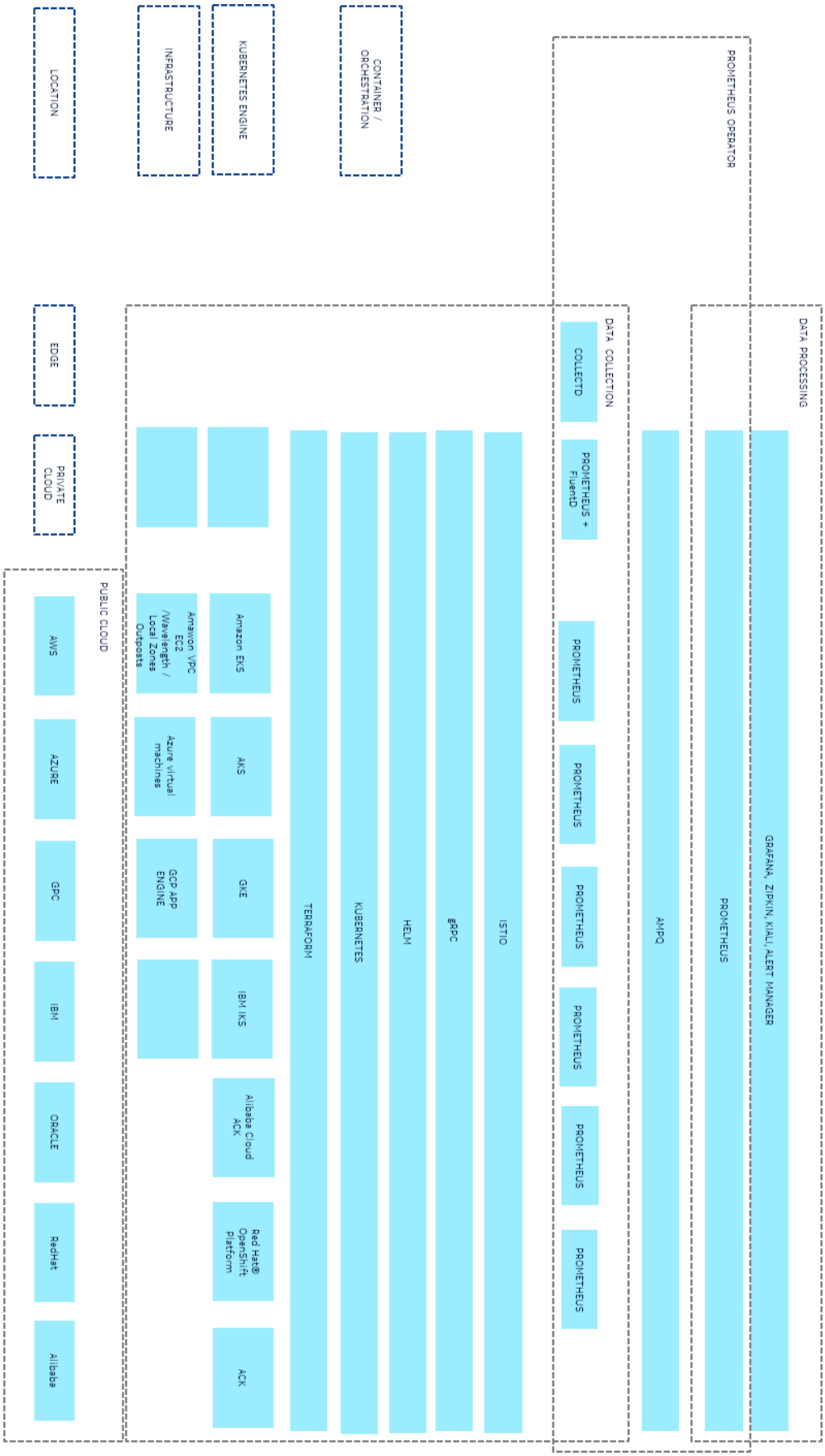

Figure 29 Proposed open-source-based architecture

5 Conclusions

As a result of this project, the following has been achieved for the company:

- Possible tools capable of performing cost-related monitoring in specified environments have been identified.
- The users and uses of cost monitoring and analysis have been identified and their needs have been defined, including specific dashboards, graphs, KPIs, and metrics that must be obtained.
- The main cost contributors have been identified, and cost-generating factors have been defined.

Based on the information in Chapter 3 and analysis made in Chapter 4, the following can be concluded:

1) Considering the state of development of the cloud, both open source and commercial alternatives should be further explored, as each offers unique opportunities. Currently, the most successful open-source projects (or ecosystems) are overseen by the Cloud Native Computing Foundation and financially backed by some of the largest corporations.

2) The ultimate tool for monitoring is Prometheus. It is an open-source tool, and its license is not in dispute with its use in commercial products. The community is extensive, support is readily available, there is an impressive number of integrations within the open-source ecosystem, and, most importantly, there is no comparable alternative.

3) The open-source ecosystem is evolving and changing at a very fast pace. This includes not only product development (or lack of it) but also relicensing. It is advisable for companies to plan for such scenarios.

References

https://www.citethisforme.com/

[1] Hybrid cloud vs. Multi-cloud [Internet]. VMware. 2021 [cited 16 September 2021]. Available from: https://www.vmware.com/topics/glossary/content/hybrid-cloud-vs-multi-cloud

[2] Geography and regions | Documentation | Google Cloud [Internet]. Google Cloud. 2021 [cited 16 September 2021]. Available from: https://cloud.google.com/docs/geography-and-regions

[3] The Needlessly Complex History of SaaS, Simplified | Process Street | Checklist, Workflow and SOP Software [Internet]. Process Street. 2021 [cited 16 September 2021]. Available from: https://www.process.st/history-of-saas/

[4] History of Cloud Computing - Interprise Software [Internet]. Interprisesoftware.com. 2021 [cited 16 September 2021]. Available from: http://www.interprisesoftware.com/cloud_history.html

[5] Jungck K, Rahman, PhD S. Cloud Computing Avoids Downfall of Application Service Providers [Internet]. International Journal of Information Technology Convergence and Services (IJITCS) Vol.1, No.3, June 2011; 2021 [cited 16 September 2021]. Available from: https://arxiv.org/ftp/arxiv/papers/1512/1512.00061.pdf

[6] Software-as-a-Service Executive Council. Software-as-a-Service; A Comprehensive Look at the Total Cost of Ownership of Software Applications [Internet]. Software & Information Industry Association; 2006 [cited 16 September 2021]. Available from: https://www.plantservices.com/assets/wp_downloads/pdf/yardstick_wp_saas_tco.pdf

[7] Miller J. Difference Between SaaS and Managed Services | BitLyft Cybersecurity [Internet]. Bitlyft.com. 2021 [cited 16 September 2021]. Available from: https://www.bitlyft.com/resources/what-is-the-difference-between-saas-and-managed-services

[8] Software as a service - Wikipedia [Internet]. En.wikipedia.org. 2021 [cited 17 September 2021]. Available from: https://en.wikipedia.org/wiki/Software_as_a_service

[9] Software as a Subscription, not as a Service [Internet]. Sia-partners.com. 2021 [cited 17 September 2021]. Available from: https://www.sia-partners.com/en/news-and-publications/from-our-experts/software-subscription-not-service

[10] Iveroth, E., Westelius, A., Petri, C.J., Olve, N.G., Coster, M., Nilsson, F.: How to differentiate by price: Proposal for a five-dimensional model. European Management Journal (2012)

[11] iaas-paas-saas [Internet]. Ibm.com. 2021 [cited 17 September 2021]. Available from: https://www.ibm.com/cloud/learn/iaas-paas-saas

[12] Public Cloud vs Private Cloud vs Hybrid Cloud | Microsoft Azure [Internet]. Azure.microsoft.com. 2021 [cited 17 September 2021]. Available from: https://azure.microsoft.com/en-us/overview/what-are-private-public-hybrid-clouds/

[13] Definition of Community Cloud - Gartner Information Technology Glossary [Internet]. Gartner. 2021 [cited 17 September 2021]. Available from: https://www.gartner.com/en/information-technology/glossary/community-cloud

[14] Sting. Public Cloud Services Comparison | @Sting (atSting.com) [Internet]. Atsting.com. 2021 [cited 17 September 2021]. Available from: https://www.atsting.com/archives/1580

[15] Production-Grade Container Orchestration [Internet]. Kubernetes. 2021 [cited 19 October 2021]. Available from: https://kubernetes.io/

[16] Swarm mode overview [Internet]. Docker Documentation. 2021 [cited 19 October 2021]. Available from: https://docs.docker.com/engine/swarm/

[17] Nomad vs. Yarn vs. Kubernetes vs. Borg vs. Mesos vs… you name it! [Internet]. Medium. 2021 [cited 19 October 2021]. Available from: https://medium.com/@arsenyspb/nomad-vs-yarn-vs-kubernetes-vs-borg-vs-mesos-vs-you-name-it-7f15a907ece2

[18] Fully Managed Container Solution – Amazon Elastic Container Service (Amazon ECS) - Amazon Web Services [Internet]. Amazon Web Services, Inc. 2021 [cited 19 October 2021]. Available from: https://aws.amazon.com/ecs/

[19] Mirantis Kubernetes Engine | Formerly Docker Enterprise | Mirantis [Internet]. Mirantis | Ship Code Faster. 2021 [cited 19 October 2021]. Available from: https://www.mirantis.com/software/mirantis-kubernetes-engine/

[20] Kubernetes - Google Kubernetes Engine (GKE) | Google Cloud [Internet]. Google Cloud. 2021 [cited 19 October 2021]. Available from: https://cloud.google.com/kubernetes-engine

[21] Red Hat OpenShift Container Platform [Internet]. 2021 [cited 19 October 2021]. Available from: https://www.redhat.com/en/technologies/cloud-computing/openshift/container-platform

[22] Azure Kubernetes Service (AKS) | Microsoft Azure [Internet]. Azure.microsoft.com. 2021 [cited 19 October 2021]. Available from: https://azure.microsoft.com/en-us/services/kubernetes-service/

[23] Container Service for Kubernetes - Alibaba Cloud [Internet]. AlibabaCloud. 2021 [cited 19 October 2021]. Available from: https://www.alibabacloud.com/product/kubernetes

[24] An Introduction to Metrics, Monitoring, and Alerting | DigitalOcean [Internet]. DigitalOcean. 2021 [cited 19 October 2021]. Available from: https://www.digitalocean.com/community/tutorials/an-introduction-to-metrics-monitoring-and-alerting

[25] What Is Service-Oriented Architecture? [Internet]. Medium. 2021 [cited 19 October 2021]. Available from: https://medium.com/@SoftwareDevelopmentCommunity/what-is-service-oriented-architecture-fa894d11a7ec

[26] SOA vs. Microservices: What's the Difference? [Internet]. Ibm.com. 2021 [cited 19 October 2021]. Available from: https://www.ibm.com/cloud/blog/soa-vs-microservices

[27] Education I. Service Oriented Architecture [Internet]. Ibm.com. 2021 [cited 21 October 2021]. Available from: https://www.ibm.com/cloud/learn/soa

[28] Why You Can't Talk About Microservices Without Mentioning Netflix [Internet]. SmartBear.com. 2015 [cited 21 October 2021]. Available from: https://smartbear.com/blog/why-you-cant-talk-about-microservices-without-ment/

[29] Service meshes in a microservices architecture [Internet]. Google Cloud. 2021 [cited 21 October 2021]. Available from: https://cloud.google.com/architecture/service-meshes-in-microservices-architecture

[30] What's a service mesh? [Internet]. 2018 [cited 21 October 2021]. Available from: https://www.redhat.com/en/topics/microservices/what-is-a-service-mesh

[31] Load Balancer Service type for Kubernetes [Internet]. Medium. 2020 [cited 21 October 2021]. Available from: https://medium.com/avmconsulting-blog/external-ip-service-type-for-kubernetes-ec2073ef5442

[32] Microservices with Kubernetes and Docker - Piotr's TechBlog [Internet]. Piotr's TechBlog. 2017 [cited 21 October 2021]. Available from: https://piotrminkowski.com/2017/03/31/microservices-with-kubernetes-and-docker/

[33] MSV J. Deploy a Multicluster Ingress on Google Kubernetes Engine - The New Stack [Internet]. The New Stack. 2018 [cited 21 October 2021]. Available from: https://thenewstack.io/deploy-a-multicluster-ingress-on-google-kubernetes-engine/

[34] Service Catalog [Internet]. Kubernetes. 2021 [cited 21 October 2021]. Available from: https://kubernetes.io/docs/concepts/extend-kubernetes/service-catalog/

[35] Single-Tenant vs Multi-Tenant: Which Path Should You Take? | OroCommerce [Internet]. OroCommerce. 2021 [cited 21 October 2021]. Available from: https://oroinc.com/b2b-ecommerce/blog/single-tenant-vs-multi-tenant/

[36] Madsen C. The 10 best KPIs for every SaaS business [Internet]. Plecto. 2019 [cited 23 October 2021]. Available from: https://www.plecto.com/blog/sales-performance/10-saas-kpis-you-should-focus/

[37] SaaS Metrics & KPIs [Internet]. Klipfolio.com. 2021 [cited 23 October 2021]. Available from: https://www.klipfolio.com/resources/kpi-examples/saas-metrics

[38] Doubling down on open, Part II [Internet]. Elastic Blog. 2021 [cited 24 October 2021]. Available from: https://www.elastic.co/blog/licensing-change

[39] Elasticsearch and Kibana are now business risks [Internet]. 2021 [cited 24 October 2021]. Available from:

https://anonymoushash.vmbrasseur.com/2021/01/14/elasticsearch-and-kibana-are-now-business-risks

[40] Signal Messenger is no longer open source - World Today News [Internet]. World Today News. 2021 [cited 24 October 2021]. Available from: https://www.world-today-news.com/signal-messenger-is-no-longer-open-source/

[41] [Internet]. Reddit.com. 2017 [cited 24 October 2021]. Available from: https://www.reddit.com/r/changelog/comments/6xfyfg/an_update_on_the_state_of_the_redditreddit_and/

[42] Mapbox GL JS Is No Longer Open Source [Internet]. WP Tavern. 2020 [cited 24 October 2021]. Available from: https://wptavern.com/mapbox-gl-js-is-no-longer-open-source

[43] Not So Open Any More: Elasticsearch Relicensing and Implications for Open Source Search [Internet]. reworked.co. 2021 [cited 24 October 2021]. Available from: https://www.reworked.co/knowledge-findability/not-so-open-any-more-elasticsearch-relicensing-and-implications-for-open-source-search/

[44] MongoDB now released under the Server Side Public License | MongoDB Blog [Internet]. MongoDB. 2018 [cited 24 October 2021]. Available from: https://www.mongodb.com/blog/post/mongodb-now-released-under-the-server-side-public-license

[45] Licenses & Standards | Open Source Initiative [Internet]. Opensource.org. 2021 [cited 24 October 2021]. Available from: https://opensource.org/licenses

[46] Why We're Relicensing CockroachDB [Internet]. Cockroach Labs. 2019 [cited 24 October 2021]. Available from: https://www.cockroachlabs.com/blog/oss-relicensing-cockroachdb/

[47] Redis Labs' Modules License Changes - Redis [Internet]. Redis. 2019 [cited 24 October 2021]. Available from: https://redis.com/blog/redis-labs-modules-license-changes/

[48] Building a self-sustaining open-source business in the cloud era [Internet]. Timescale Blog. 2020 [cited 24 October 2021]. Available from: https://blog.timescale.com/blog/building-open-source-business-in-cloud-era-v2/

[49] Graylog v4.0 Licensing SSPL | Graylog [Internet]. Graylog.org. 2020 [cited 24 October 2021]. Available from: https://www.graylog.org/post/graylog-v4-0-licensing-sspl

[50] License Changes for Confluent Platform [Internet]. 2018 [cited 24 October 2021]. Available from: https://www.confluent.io/blog/license-changes-confluent-platform/

[51] Prometheus - Monitoring system & time series database [Internet]. Prometheus.io. 2021 [cited 24 October 2021]. Available from: https://prometheus.io/

[52] GitHub - prometheus/prometheus: The Prometheus monitoring system and time series database. [Internet]. GitHub. 2021 [cited 24 October 2021]. Available from: https://github.com/prometheus/prometheus

[53] Overview | Prometheus [Internet]. Prometheus.io. 2021 [cited 24 October 2021]. Available from: https://prometheus.io/docs/introduction/overview/

[54] Exporters and integrations | Prometheus [Internet]. Prometheus.io. 2021 [cited 24 October 2021]. Available from: https://prometheus.io/docs/instrumenting/exporters/

[55] Alertmanager | Prometheus [Internet]. Prometheus.io. 2021 [cited 24 October 2021]. Available from: https://prometheus.io/docs/alerting/latest/alertmanager/

[56] GitHub - prometheus/alertmanager: Prometheus Alertmanager [Internet]. GitHub. 2021 [cited 24 October 2021]. Available from: https://github.com/prometheus/alertmanager

[57] Federation | Prometheus [Internet]. Prometheus.io. 2021 [cited 24 October 2021]. Available from: https://prometheus.io/docs/prometheus/latest/federation/#hierarchical-federation

[58] Monitoring Multicluster Istio with Prometheus [Internet]. Istio. 2021 [cited 24 October 2021]. Available from: https://istio.io/latest/docs/ops/configuration/telemetry/monitoring-multicluster-prometheus/

[59] FAQ — Graphite 1.2.0 documentation [Internet]. Graphite.readthedocs.io. 2021 [cited 24 October 2021]. Available from: https://graphite.readthedocs.io/en/latest/faq.html

[60] Comparison to alternatives | Prometheus [Internet]. Prometheus.io. 2021 [cited 24 October 2021]. Available from: https://prometheus.io/docs/introduction/comparison/

[61] Monitoring Kubernetes with Graphite | MetricFire Blog [Internet]. Metricfire.com. 2021 [cited 24 October 2021]. Available from: https://www.metricfire.com/blog/monitoring-kubernetes-with-graphite/

[62] GitHub - statsd/statsd: Daemon for easy but powerful stats aggregation [Internet]. GitHub. 2021 [cited 24 October 2021]. Available from: https://github.com/statsd/statsd

[63] Graphite [Internet]. Graphiteapp.org. 2021 [cited 24 October 2021]. Available from: https://graphiteapp.org/

[64] Graphite Project [Internet]. GitHub. 2021 [cited 24 October 2021]. Available from: https://github.com/graphite-project

[65] Thanos [Internet]. Thanos.io. 2021 [cited 24 October 2021]. Available from: https://thanos.io/

[66] GitHub - thanos-io/thanos: Highly available Prometheus setup with long term storage capabilities. A CNCF Incubating project. [Internet]. GitHub. 2021 [cited 24 October 2021]. Available from: https://github.com/thanos-io/thanos

[67] Pracucci M. How the Cortex and Thanos projects collaborate to make scaling Prometheus better for all [Internet]. Grafana. 2020 [cited 24 October 2021]. Available from: https://grafana.com/blog/2020/07/16/how-the-cortex-and-thanos-projects-collaborate-to-make-scaling-prometheus-better-for-all/

[68] Cortex [Internet]. Cortex. 2021 [cited 24 October 2021]. Available from: https://cortexmetrics.io/

[69] Cortex [Internet]. GitHub. 2021 [cited 24 October 2021]. Available from: https://github.com/cortexproject

[70] Wilkie T. [PromCon Recap] Two Households, Both Alike in Dignity: Cortex and Thanos [Internet]. Grafana. 2019 [cited 24 October 2021]. Available from: https://grafana.com/blog/2019/11/21/promcon-recap-two-households-both-alike-in-dignity-cortex-and-thanos/

[71] Perkins L. Cortex: a multi-tenant, horizontally scalable Prometheus-as-a-Service | Cloud Native Computing Foundation [Internet]. Cloud Native Computing Foundation. 2018 [cited 24 October 2021]. Available from: https://www.cncf.io/blog/2018/12/18/cortex-a-multi-tenant-horizontally-scalable-prometheus-as-a-service/

[72] Cortex Architecture [Internet]. Cortex. 2021 [cited 24 October 2021]. Available from: https://cortexmetrics.io/docs/architecture/

[73] Zabbix :: The Enterprise-Class Open Source Network Monitoring Solution [Internet]. Zabbix.com. 2021 [cited 24 October 2021]. Available from: https://www.zabbix.com/

[74] GitHub - zabbix/zabbix: Real-time monitoring of IT components and services, such as networks, servers, VMs, applications and the cloud. [Internet]. GitHub. 2021 [cited 24 October 2021]. Available from: https://github.com/zabbix/zabbix

[75] What's in Zabbix 5.0 LTS [Internet]. Zabbix.com. 2021 [cited 24 October 2021]. Available from: https://www.zabbix.com/cz/whats_new_5_0

[76] PostgreSQL monitoring using Zabbix Agent 2: easy and extensible [Internet]. Postgrespro.com. 2020 [cited 24 October 2021]. Available from: https://postgrespro.com/blog/pgsql/5967895

[77] Campion N. Prometheus vs. Zabbix | MetricFire Blog [Internet]. Metricfire.com. 2020 [cited 24 October 2021]. Available from: https://www.metricfire.com/blog/prometheus-vs-zabbix/#span-stylefontweight-400Zabbix-short-overviewspanspan-stylefontweight-400span

[78] Lambert D. Multi-tenant monitoring: how to achieve that using free Zabbix open-source monitoring software [Internet]. Zabbix Blog. 2020 [cited 24 October 2021]. Available from: https://blog.zabbix.com/multi-tenant-monitoring-how-to-achieve-that-using-free-zabbix-open-source-monitoring-software/12024/

[79] Telegraf [Internet]. Influxdata. 2021 [cited 24 October 2021]. Available from: https://www.influxdata.com/time-series-platform/telegraf/

[80] GitHub - influxdata/telegraf: The plugin-driven server agent for collecting & reporting metrics. [Internet]. GitHub. 2021 [cited 24 October 2021]. Available from: https://github.com/influxdata/telegraf

[81] Elasticsearch: The Official Distributed Search & Analytics Engine | Elastic [Internet]. Elastic. 2021 [cited 24 October 2021]. Available from: https://www.elastic.co/elasticsearch/

[82] GitHub - elastic/elasticsearch: Free and Open, Distributed, RESTful Search Engine [Internet]. GitHub. 2021 [cited 24 October 2021]. Available from: https://github.com/elastic/elasticsearch

[83] Logstash: Collect, Parse, Transform Logs | Elastic [Internet]. Elastic. 2021 [cited 24 October 2021]. Available from: https://www.elastic.co/logstash/

[84] GitHub - elastic/logstash: Logstash - transport and process your logs, events, or other data [Internet]. GitHub. 2021 [cited 24 October 2021]. Available from: https://github.com/elastic/logstash

[85] [Internet]. Grafana. 2021 [cited 25 October 2021]. Available from: https://grafana.com/

[86] Grafana Labs [Internet]. GitHub. 2021 [cited 25 October 2021]. Available from: https://github.com/grafana/

[87] Kibana: Explore, Visualize, Discover Data | Elastic [Internet]. Elastic. 2021 [cited 25 October 2021]. Available from: https://www.elastic.co/kibana

[88] GitHub - elastic/kibana: Your window into the Elastic Stack [Internet]. GitHub. 2021 [cited 25 October 2021]. Available from: https://github.com/elastic/kibana

[89] Kibana Alerting: Alerts & Actions for Elasticsearch data [Internet]. Elastic. 2021 [cited 25 October 2021]. Available from: https://www.elastic.co/what-is/kibana-alerting

[90] Chronograf [Internet]. Influxdata. 2021 [cited 25 October 2021]. Available from: https://www.influxdata.com/time-series-platform/chronograf/

[91] GitHub - influxdata/chronograf: Open source monitoring and visualization UI for the TICK stack [Internet]. GitHub. 2021 [cited 25 October 2021]. Available from: https://github.com/influxdata/chronograf

[92] Create Chronograf dashboards [Internet]. Influxdata. 2021 [cited 25 October 2021]. Available from: https://docs.influxdata.com/chronograf/v1.9/guides/create-a-dashboard/

[93] Weave Scope [Internet]. Weave.works. 2021 [cited 25 October 2021]. Available from: https://www.weave.works/oss/scope/

[94] GitHub - weaveworks/scope: Monitoring, visualisation & management for Docker & Kubernetes [Internet]. GitHub. 2021 [cited 25 October 2021]. Available from: https://github.com/weaveworks/scope

[95] Osborne C. Weave Scope is now being exploited in attacks against cloud environments | ZDNet [Internet]. ZDNet. 2021 [cited 25 October 2021]. Available from: https://www.zdnet.com/article/weave-scope-is-now-being-exploited-in-attacks-against-cloud-environments/

[96] Issue #3046 · weaveworks/scope [Internet]. GitHub. 2018 [cited 25 October 2021]. Available from: https://github.com/weaveworks/scope/issues/3046

[97] InfluxDB 1.x [Internet]. 2021 [cited 25 October 2021]. Available from: https://www.influxdata.com/time-series-platform/

[98] ELK Stack: Elasticsearch, Logstash, Kibana [Internet]. Elastic.co. 2021 [cited 25 October 2021]. Available from: https://www.elastic.co/what-is/elk-stack

[99] The ELK stack [Internet]. Amazon Web Services, Inc. 2021 [cited 25 October 2021]. Available from: https://aws.amazon.com/elasticsearch-service/the-elk-stack/

[100] Elastic Cloud (Elasticsearch managed service) [Internet]. Google Cloud Platform. 2021 [cited 25 October 2021]. Available from: https://console.cloud.google.com/marketplace/product/endpoints/elasticsearch-service.gcpmarketplace.elastic.co

[101] Elastic on Azure | Microsoft Azure [Internet]. Azure.microsoft.com. 2021 [cited 25 October 2021]. Available from: https://azure.microsoft.com/en-us/overview/linux-on-azure/elastic/

[102] SIEM ELK Stack | What is SIEM ELK Stack - HKR Trainings [Internet]. Hkrtrainings.com. 2021 [cited 25 October 2021]. Available from: https://hkrtrainings.com/siem-elk-stack

[103] Gungor U. Kubernetes Monitoring and Logging Solution: EMG and EFK Stack (Part 1) [Internet]. Medium. 2020 [cited 25 October 2021]. Available from: https://itnext.io/kubernetes-monitoring-and-logging-solution-emg-and-efk-stack-part-1-8aa58339e7a4

[104] Peri N. Fluentd vs Logstash: A Comparison of Log Collectors [Internet]. Logz.io. 2020 [cited 25 October 2021]. Available from: https://logz.io/blog/fluentd-logstash/

[105] GitHub - cdwv/efk-stack-helm: Helm chart to deploy a working logging solution using the ElasticSearch - Fluentd - Kibana stack on Kubernetes [Internet]. GitHub. 2021 [cited 25 October 2021]. Available from: https://github.com/cdwv/efk-stack-helm

[106] Hanif J. How To Set Up an Elasticsearch, Fluentd and Kibana (EFK) Logging Stack on Kubernetes | DigitalOcean [Internet]. DigitalOcean. 2020 [cited 25 October 2021]. Available from:

https://www.digitalocean.com/community/tutorials/how-to-set-up-an-elasticsearch-fluentd-and-kibana-efk-logging-stack-on-kubernetes

[107] Jagadeesalu B. ELK/EFK compare with Splunk [Internet]. Medium. 2019 [cited 25 October 2021]. Available from: https://medium.com/@balajijk/elk-efk-compare-with-splunk-4c18fc362fd6

[108] aesher9o1. EFK Stack: Elasticsearch, Fluentd and Kibana on Docker [Internet]. Medium. 2020 [cited 25 October 2021]. Available from: https://aesher9o1.medium.com/efk-stack-elasticsearch-fluentd-and-kibana-on-docker-be60597fa99

[109] Introducing the World's First Modern Cloud-Based SecOps Platform: Splunk Security Cloud [Internet]. Splunk-Blogs. 2021 [cited 25 October 2021]. Available from: https://www.splunk.com/en_us/blog/security/introducing-the-world-s-first-modern-cloud-based-secops-platform-splunk-security-cloud.html

[110] The world's # 1 APM solution [Internet]. AppDynamics. 2021 [cited 25 October 2021]. Available from: https://www.appdynamics.com/

[111] Kubernetes monitoring | Dynatrace [Internet]. Dynatrace. 2021 [cited 25 October 2021]. Available from: https://www.dynatrace.com/technologies/kubernetes-monitoring/

[112] Instana - Enterprise Observability and APM for Cloud-Native Applications [Internet]. Instana. 2021 [cited 25 October 2021]. Available from: https://www.instana.com/

[113] Sensu | Observability Pipeline [Internet]. Sensu.io. 2021 [cited 25 October 2021]. Available from: https://sensu.io/

[114] Comprehensive Automation for DevOps [Internet]. 2021 [cited 25 October 2021]. Available from: https://puppet.com/

[115] Azure Monitor | Microsoft Azure [Internet]. Azure.microsoft.com. 2021 [cited 25 October 2021]. Available from: https://azure.microsoft.com/en-us/services/monitor/

[116] Amazon CloudWatch - Application and Infrastructure Monitoring [Internet]. Amazon Web Services, Inc. 2021 [cited 25 October 2021]. Available from: https://aws.amazon.com/cloudwatch/

[117] Cloud Monitoring | Google Cloud [Internet]. Google Cloud. 2021 [cited 25 October 2021]. Available from: https://cloud.google.com/monitoring/

[118] CloudZero: Cloud Cost Intelligence [Internet]. Cloudzero.com. 2021 [cited 26 October 2021]. Available from: https://www.cloudzero.com/

[119] Cloud Cost Management [Internet]. Harness. 2021 [cited 26 October 2021]. Available from: https://harness.io/platform/cloud-cost-management/

[120] The Cloud Cost Management Buyer's Guide [Internet]. Harness. 2021 [cited 26 October 2021]. Available from: https://harness.io/ccm-buyers-guide-ebook/?utm_source=Internal&utm_medium=drift&utm_campaign=ccm-targeted-pb#flex-columns-block-form

[121] Trusted Technology Investment Decisions | Apptio [Internet]. Apptio. 2021 [cited 26 October 2021]. Available from: https://www.apptio.com/

[122] Total Visibility Cloud Management Platform | CloudCheckr [Internet]. CloudCheckr. 2021 [cited 26 October 2021]. Available from: https://cloudcheckr.com/

[123] Cloud Financial Management [Internet]. CloudHealth by VMware. 2021 [cited 26 October 2021]. Available from: https://www.cloudhealthtech.com/solutions/cloud-financial-management

[124] Deep Cloud Cost Optimization [Internet]. Computesoftware.com. 2021 [cited 26 October 2021]. Available from: https://www.computesoftware.com/

[125] ParkMyCloud - Reduce Cloud Costs Automatically with ParkMyCloud [Internet]. ParkMyCloud. 2021 [cited 26 October 2021]. Available from: https://www.parkmycloud.com/

[126] AWS Budgets – Amazon Web Services [Internet]. Amazon Web Services, Inc. 2021 [cited 26 October 2021]. Available from: https://aws.amazon.com/ru/aws-cost-management/aws-budgets/

[127] Cloud Billing documentation | Google Cloud [Internet]. Google Cloud. 2021 [cited 26 October 2021]. Available from: https://cloud.google.com/billing/docs

[128] Cloud Cost Management | Microsoft Azure [Internet]. Azure.microsoft.com. 2021 [cited 26 October 2021]. Available from: https://azure.microsoft.com/en-us/services/cost-management/#overview

[129] Kubecost | Kubernetes cost monitoring and management [Internet]. Kubecost.com. 2021 [cited 26 October 2021]. Available from: https://www.kubecost.com

[130] Integrated visibility & automation for cloud optimization [Internet]. Spot | Cloud Analyzer. 2021 [cited 26 October 2021]. Available from: https://spot.io/products/cloud-analyzer/

[131] Application Resource Management for Digital Business Transformation [Internet]. Turbonomic.com. 2021 [cited 26 October 2021]. Available from: https://www.turbonomic.com/

[132] Yotascale - Dynamic Cloud Cost Management [Internet]. Yotascale. 2021 [cited 26 October 2021]. Available from: https://yotascale.com/

[133] OKD - The Community Distribution of Kubernetes that powers Red Hat OpenShift. [Internet]. Okd.io. 2021 [cited 26 October 2021]. Available from: https://www.okd.io/

[134] AWS Pricing Calculator [Internet]. Calculator.aws. 2021 [cited 26 October 2021]. Available from: https://calculator.aws/#/addService

[135] Google Cloud Platform Pricing Calculator [Internet]. Google Cloud. 2021 [cited 26 October 2021]. Available from: https://cloud.google.com/products/calculator

[136] Pricing Calculator | Microsoft Azure [Internet]. Azure.microsoft.com. 2021 [cited 26 October 2021]. Available from: https://azure.microsoft.com/en-us/pricing/calculator/

[137] IBM Cloud cost estimator [Internet]. Ibm.com. 2021 [cited 26 October 2021]. Available from: https://www.ibm.com/cloud/cloud-calculator

[138] Price Calculator [Internet]. Alibabacloud.com. 2021 [cited 26 October 2021]. Available from: https://www.alibabacloud.com/pricing-calculator#/commodity/vm_intl

[139] Can private cloud be cheaper than public cloud? [Internet]. 451 RESEARCH; 2017 [cited 26 October 2021]. Available from: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/products/vrealize-suite/vmware-paper1-can-private-cloud-be-cheaper-than-public-cloud.pdf

[140] Monitoring that understands business [Internet]. Anodot.com. 2021 [cited 26 October 2021]. Available from: https://www.anodot.com/

[141] CloudZero [Internet]. Capterra. 2021 [cited 26 October 2021]. Available from: https://www.capterra.ie/software/200498/cloudzero

[142] Budgets and Forecasts - Apptio Cloudability [Internet]. Apptio. 2021 [cited 26 October 2021]. Available from: https://www.apptio.com/products/cloudability/budgets-forecasts/

[143] Cloud Management FAQs [Internet]. ParkMyCloud. 2021 [cited 26 October 2021]. Available from: https://www.parkmycloud.com/faqs/

[144] Currently used pricing models in the cloud industry [Internet]. 2021 [cited 26 October 2021]. Available from: https://www.researchgate.net/figure/Currently-used-pricing-models-in-the-cloud-industry_fig2_270958130

[145] What is Technology Business Management? TBM Explained [Internet]. BMC Blogs. 2021 [cited 26 October 2021]. Available from: https://www.bmc.com/blogs/tbm-technology-business-management/

[146] Analytics - Overview of Amazon Web Services [Internet]. Docs.aws.amazon.com. 2021 [cited 26 October 2021]. Available from: https://docs.aws.amazon.com/whitepapers/latest/aws-overview/analytics.html

[147] Compare AWS and Azure services to Google Cloud [Internet]. Google Cloud. 2021 [cited 26 October 2021]. Available from: https://cloud.google.com/free/docs/aws-azure-gcp-service-comparison