Optimization of a Cloud Environment after Migration into AWS

Bc. Michal Bernátek

Master's thesis
2023

Tomas Bata University in Zlín
Faculty of Applied Informatics

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Michal Bernátek**
Osobní číslo:	**A21490**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Optimalizace cloudového prostředí po migraci do AWS**
Téma práce anglicky:	**Optimization of a Cloud Environment after Migration into AWS**

Zásady pro vypracování

1. Seznamte se s výchozím stavem implementace cloudového prostředí ve vaší organizaci a propojením s platformou AWS.
2. Popište procesy a analyzujte služby, které prostředí používá.
3. Navrhněte možné optimalizace jejich implementaci.
4. Realizujte návrhy implementace.
5. Vyhodnoďte náklady po implementaci specifických optimalizací.

Forma zpracování diplomové práce:	**tištěná/elektronická**
Jazyk zpracování:	**Angličtina**

Seznam doporučené literatury:

1. PIPER, Ben a David CLINTON, 2019. AWS Certified Cloud Practitioner Study Guide CLF-C01 Exam. Indianapolis: Wiley. ISBN 978-1-119-49070-8.
2. ARMSTRONG, Jeff, 2020. Migrating to AWS: Manager's Guide, Indianapolis: Wiley. ISBN 978-1-492-07424-3
3. BURNS, Brendan a VILLALBA, Eddie a STREBEL, Dave a EVENSON, Lachlan, 2019. Kubernetes Best Practices. Indianapolis: Wiley. ISBN 978-1-492-05647-8
4. PIPER, Ben a David CLINTON, 2019. AWS Certified Solutions Architect Study Guide. Second Edition. Indianapolis: Wiley. ISBN 978-1-119-50421-4.
5. PERROTT, Sara a Brett MCLAUGHLIN, 2020. AWS Certified SysOps Administrator Study Guide. Second Edition. Indianapolis: Wiley. ISBN 978-1-119-56155-2.

Vedoucí diplomové práce:	**Ing. Tomáš Dulík, Ph.D.**
	Ústav informatiky a umělé inteligence
Datum zadání diplomové práce:	**2. prosince 2022**
Termín odevzdání diplomové práce:	**26. května 2023**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
①

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

ABSTRAKT

Téma diplomové práce "Optimalizace cloudového prostředí po migraci do AWS" se zabývá procesy možné optimalizace po migraci do cloudového prostředí Amazon Web Services za účelem využití cloud nativních technologií, zrychlení aplikací, snížení nákladů a také zvýšení elasticity celého řešení. Konkrétně práce navazuje na business případ, kdy firma Papirfly AS (dříve BrandMaster AS) přesunula veškerou svojí infrastrukturu ze svého vlastního datacentra do cloudového prostředí AWS a dále v něm operuje. Cílem práce je popsat momentální stav celého řešení a všech propojení v rámci cloudového prostředí AWS a analyzovat služby, které řešení používá k běhu. Dále vyhodnotit a navrhnout možné optimalizace a některé z nich implementovat. Nakonec se práce zabývá snížením nákladů následkem zmíněných optimalizací.

Klíčové slova:

Amazon Web Services, AWS, Cloud, Cloud Computing, Optimalizace, Prostředí, Náklady, Redukce nákladů, Cloudová architektura, DevOps, Kubernetes, Škálování, Elasticita, Správa nákladů

ABSTRACT

The master's thesis "Optimization of a cloud environment after migration into AWS" investigates processes of optimizations of a cloud environment inside Amazon Web Services for purposes of cloud native technology usage, speeding up the solution running in the environment, lowering the costs, and making the solution more scalable. The thesis is a continuation of a business case where Papirfly AS (formerly known as BrandMaster AS) migrated their whole infrastructure into the AWS from their self-hosted on-premises solution and is continuing using the AWS as a primary hosting platform for its solution. The intention of the thesis is to describe the current state of the whole solution, all the applications and their connections and integrations into the AWS platform. A secondary purpose of the thesis is to evaluate and propose optimizations and implement some of them. Finally, the thesis should dive into the lowered costs connected to the optimizations mentioned.

Key words:

Amazon Web Services, AWS, Cloud, Cloud Computing, Optimization, Environment, Costs, Costs reduction, Cloud Architecture, DevOps, Kubernetes, Scaling, Elasticity, Billing

Rád bych poděkoval panu Ing. Tomáši Dulíkovi, Ph.D. za možnost zpracování tématu diplomové práce spojeného s mou osobní praxí a zaměstnáním, a také za provedení procesem přípravy a vypracování diplomové práce.

I would also like to thank the CTO of Papirfly AS, Morten Moe, for allowing me to use the Papirfly's business case again for the purpose of this thesis. I would also like to thank my great colleague, Jørgen Mølbach, who was a great support during the process of optimizing the AWS environments.

CONTENTS

INTRODUCTION

Cloud Computing is a phenomenon which is gaining popularity with software development companies in the recent years as there is more push towards empowering developers through using managed services and platforms so the developers themselves do not have to manage and maintain any physical and sometimes even logical or emulated infrastructure. There is also more need for elasticity as the global systems can go through drastic spikes during the daytime peaks and fall towards almost hibernation-like states during the nights. These states are usually extremely hard to manage on a physical infrastructure and that is where the Cloud Computing itself accelerates as for the front-facing customer the costs can essentially go to zero if the environment is set up in a way to accommodate for scaling to zero.[2]

The thesis is an exploration of a post migration environment as a continuation of a migration business case presented by Papirfly AS, formerly known as BrandMaster AS, which has migrated its whole infrastructure and operations into the cloud while going through a merger of two organizations and aligning their operations into the same environment in the cloud, specifically Amazon Web Services, which is the leader in the cloud provider and computing space since 2006. There is a huge environment in place which consists of multiple sub-environments used for diverse purposes in the development cycle and for logical separation on multiple levels. This thesis describes the state of these environments in their isolated state and the connections between them as they interact and communicate with each other. There is a description of all the cloud native components used throughout the system and how the components build the platform together with the applications developed by the Papirfly organization.

Analysis in the thesis aims to analyze narrow points throughout the system which can be optimized by using smarter techniques and/or also usage of the cloud native components or a possible usage of managed services provided by Amazon Web Services for better pricing or by making use of special deals and offers Amazon Web Services present in form of reserved or pre-purchased capacity or by using the spot market available on the AWS platform for the Cloud Computing service itself. Furthermore, the exploration of these techniques aims to provide appropriate recommendations when it comes to lowering the operation costs of the whole solution and the thesis aims to implement some of these recommendations and solutions to measure the impact on the cost of the entire environment in the end by providing concrete reports using the Amazon Web Services pricing and cost tooling.

I. THEORY

1 CLOUD COMPUTING

Cloud Computing is a popular pattern which has been gaining popularity with various companies and organizations not only in the IT space in recent years. It is interesting for diverse kinds of businesses for the sole reason of the fact that the cloud providers are offering distinct abstractions of infrastructure levels depending on the customer's needs, requirements, and financial options. The cloud providers are doing so by offering different services which can be managed on their platform. They can also offer just the basic level of the infrastructure as a service but can go up to the highest level where the whole software is being offered as a service by the cloud provider to the customer paying for the selected solution.

The major share of the cloud computing market space is still divided between three tech giants even after a few decades the IT world has invested into the cloud computing space. The giants being Amazon Web Services, Microsoft Azure, and Google Cloud Platform, where Amazon Web services was leading with 34% of the whole cloud computing space according to reports from the end of third quarter of 2022. The second giant trailing closely behind Amazon Web Services, Microsoft Azure, was taking up 21% of the whole cloud computing market space at the end of the third quarter of 2022. The third tech giant Google Cloud Platform was spread throughout 11% of the cloud computing market at the end of the third quarter of 2022, being followed by various smaller cloud providers from all around the world occupying 5% or less of the market space per cloud provider. Some of the cloud providers might be Alibaba Cloud, IBM Cloud, Sales Force, Tencent Cloud and Oracle Cloud. The overall revenue generated by the cloud providers and operators in the previous 12 months prior to September 2022 was calculated to be close to 217 billion dollars which brings the revenue for this market space up by approximately 68% when compared to reports which came out two years ago in 2020 and were reported by the Synergy Research Group.[1]

When it comes to the advantages of cloud computing and cloud providers in general, companies and organizations are choosing this way of hosting their environments for two main reasons, Elasticity and Pricing, which are interconnected in the cloud. It could be merged into "Elastic Pricing" as the most standard pricing model which every cloud provider offers are referred to as "on-demand" or "pay as you go," where the customer pays only for resources being used in the environment. There are of course flat prices being billed for resources running per hour but even those can be scaled down to zero during the night, for example. These “on-demand” or “pay as you go” billing models are offered by most of the cloud providers as they provide the customer with the most flexibility and elasticity for their environments during peak loads, unexpected traffic or other edge situations which are almost impossible to predict in the real world. There are also several levels of how the cloud providers usually bill their customers. The most common way of how they bill the customer is price propagation per hour which is being used on most of the basic resources or resources which are hard to measure otherwise, some of the components might include virtual servers or networking components which are most commonly just virtual servers under the hood anyway. The second way might be more specific to cloud storage as the customer is usually billed per GB of used space. These quite essential models are commonly overshadowed by more beneficial billing models for the customer for example the AWS Lambda platform is being billed for each microsecond a function is being executed so we can essentially say every micro-optimization in the code of the running function counts towards a better pricing.[2]

Some kind of level separation can be also talked about on the cloud computing level as the cloud providers are offering a diverse portfolio of services which can be hosted and configured on a bunch of various levels, some of them on more than one of those levels. Most essential decision being made when looking into what kind of level of abstraction is the best for the organization comes down to how much qualified personnel can the organization assign to the given service they are trying to use. There are also levels of abstractions which cut out the operation of all the underlying infrastructure. The relative levels split into a structure where the customer has to take care of the physical infrastructure and only has to ask for what infrastructure is wanted in their solution and the customer then gets it in a virtualized form which then the customer proceeds to build the solution with and continues to manage its configuration, nothing more, this scheme of hosting is most commonly referred to as Infrastructure as a Service, shortly IaaS. Next level of the cloud abstraction is often called Platform as a Service, or PaaS, where the customer does not have to think about the actual act of provisioning the right software. The cloud provider automatically provisions the infrastructure needed based on some usually very vague and simplified specification or even a simplified user interface with simple flows or wizards which lead the customer through the entire process of instantiating a server with the needed configuration and software provided by them. The next level of abstraction splits into at least three categories where the given categories are not explicitly defined nor being referred to in a standard way. All the categories could be referred to as Serverless as the abstraction usually takes away the traditional sense of a server running underneath the applications and is solely focused on running the application without thinking much of where the application is being executed as long as its runtime requirements are being satisfied. One of these categories can be referred to as Container as a Service, where the cloud providers usually provide some kind of proprietary or open-source container solution being run and managed by the cloud provider and the customer gets only certain parts of the application programmable interface to work with so they could specify the deployment procedure for the containerized application onto the managed container platform itself. The other category is referred to as Function as a Service, where the customer usually just supplies the cloud provider with the source code of the application and the Function as a Service platform deploys the code in its own preferred way, so the customer does not have to worry about any underlying container runtime information and specifics. The example of this can be seen on Amazon Web Services with their AWS Lambda service which is using Firecracker to spin up micro virtual machines on-demand and manages the lifecycle of the function executions.[8]

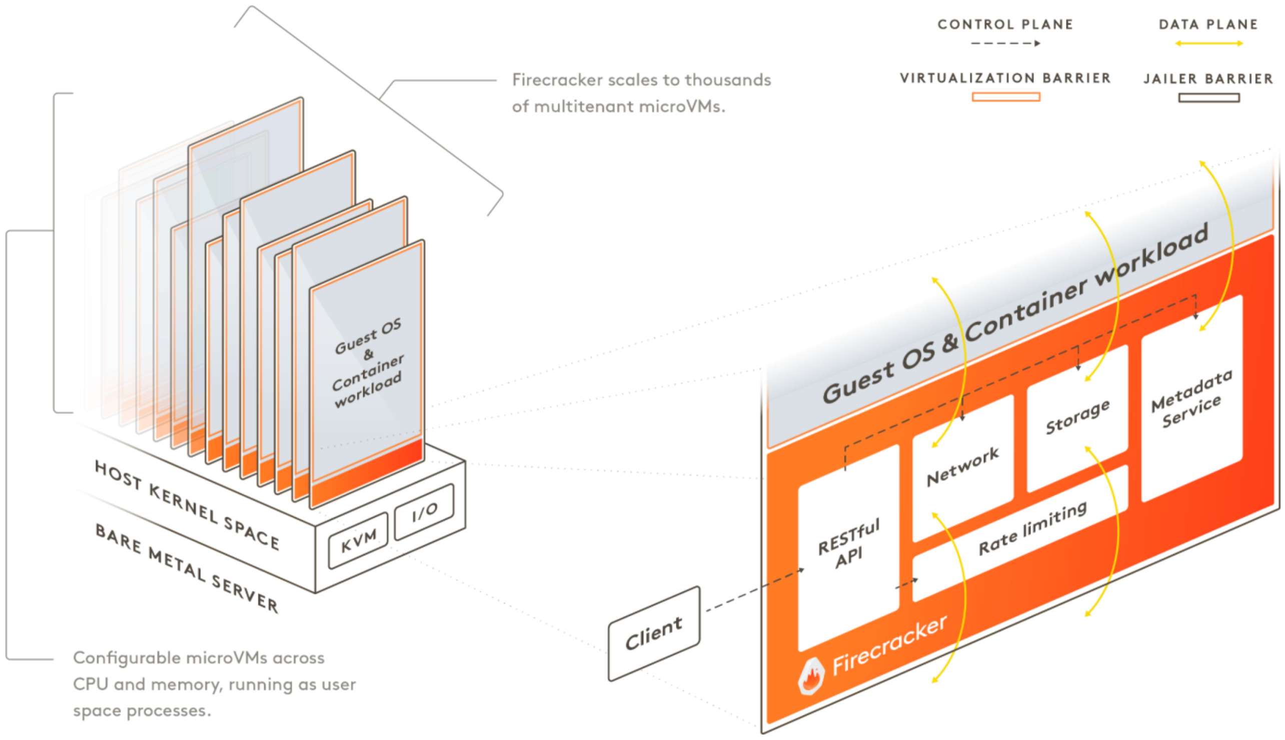
Firecracker scales to thousands of multitenant microVMs.
CONTROL PLANE
DATA PLANE
VIRTUALIZATION BARRIER
JAILER BARRIER
Guest OS & Container workload
Host Kernel Space
Bare Metal Server
KVM
I/O
Configurable microVMs across CPU and memory, running as user space processes.
Guest OS & Container workload
RESTful API
Network
Storage
Metadata Service
Rate limiting
Client
Firecracker

Figure 1 – How Firecracker works [8]

Other example can be Cloudflare's JavaScript/Web Assembly Runtime workers which is powering the Cloudflare Workers service where the user can deploy the application code and execute it on the edge and actually introduces several added terms into the serverless space as nano services and isolates.[9]

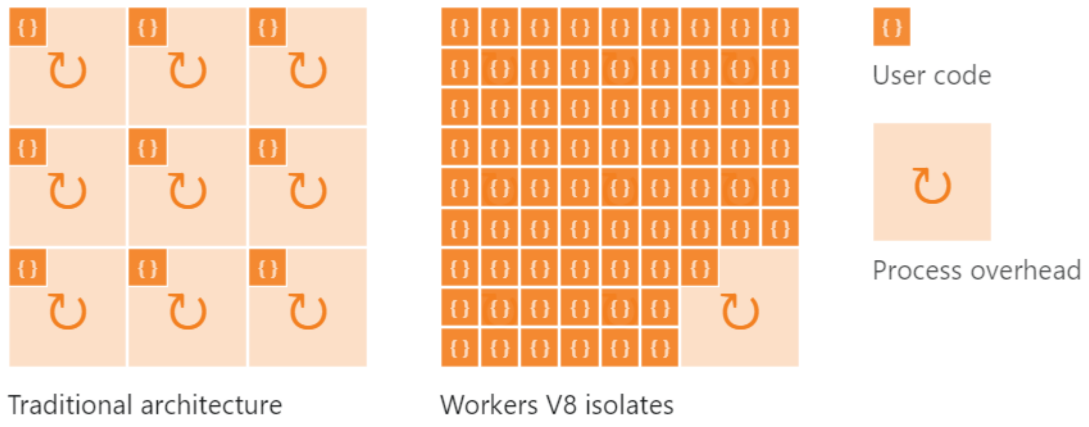

Figure 2 – Cloudflare workers architecture [9]

The last but not least form of hosting an environment is called Software as a Service where the end customer is paying for the whole operation of running software and just presented with the user interface of the application itself, everything else from the underlying infrastructure to writing the code is managed by the cloud provider and this scheme is usually used within various software companies which utilize the cloud providers for their expertise and built more products upon the platforms cloud providers provide.[11]

Significant division of the cloud providers can be determined also on the tenancy of the given cloud computing environment as the customers can choose from various approaches where they can either share the computation resources in a multitenant manner which the other users of the given cloud environment or they can have completely dedicated a separated environment just for their computational operations and workloads. Depending on the service being used by the customer and the way the cloud provider implemented the given service these two approaches can be of course combined giving the end customer and user of the service more flexibility over the environment where they want to execute their workloads. The main separation of the tenancy level would split the cloud environments into three ways of running workloads, Public Cloud Computing, Private Cloud Computing and Hybrid Cloud Computing. Public Cloud Computing is, as the name suggests, a way where the workloads are running in a multi-tenant environment and the computation resources are being shared between multiple customers and their workloads are being executed on the same underlying hardware at one given point in time. This way of running workloads can of course lead to security concerns and to even more compliance reasons as the runtime of the applications cannot really be determined by the customer of the cloud provider. Private Cloud Computing on the other hand, as the previous explanation suggests, is the total opposite of the Public Cloud Computing as it is being chosen as the only way of meeting various regulatory compliance requirements and reasons or just to meet requirements given by the customer to fulfill contractual obligations and agreements signed with their clients. The customer of the cloud provider spins up an environment in a completely single tenant way and the resources are used for computational workloads only for this one given customer, and thus achieving complete control over the environment, and therefore the compliance requirements can be met this way. Last of the three approaches is Hybrid Cloud Computing and it can be described as simple as a combination of the Public and Private Cloud Computing into one environment which is then working in a hybrid way where some of the components are running in a single tenant way, being dedicated to the one and only customer, and others are running in a multitenant way where they are being shared with all the other users of the given cloud provider.[10]

With the rise of the cloud computing as the preferred way of running workloads in general there has been an enormous amount of competitiveness the in the cloud computing space and thus it is natural the cloud providers are racing against each other to close contracts with customers by providing better pricing or better services the others might not provider or might not provide with the same quality as they do. A fourth approach was born and is called Multi Cloud Computing where the customer combines various cloud providers and specifically chooses which services from the given cloud providers they want to use and then interconnects them between the chosen cloud environments, usually chasing either better functionality or quality or better prices for the given service used. The most essential benefit of this approach is of course the huge flexibility which comes out of this solution as the end user can choose whatever cloud provider they want, for whatever reason they like, and they are also avoiding a very discussed, so called, vendor lock-in, where the customer of the cloud provider essentially becomes dependent on the given cloud provider and it becomes very difficult to migrate away from the given cloud provider in case of a breach of contracts or just a simple better offer from any of the other cloud providers in the cloud computing space. The often-overlooked disadvantage is the simple fact of the environment being hard to maintain and becoming complex if a lot of the cloud providers are used and interconnected.[2]

2 CLOUD PROVIDERS

A cloud provider, also referred to as Cloud Service provider, shortly CSP, is typically an information technology company or organization which is solely aiming to provide abstraction of infrastructure and its provisioning and management to customers which are willing to pay for these services. They are commonly delivered in an on-demand fashion over the internet in the form of a user interface and a publicly exposed API which the customer can use to manipulate different resources of the cloud provider on their platform.

Cloud providers offer three types of services depending on the level of abstraction the customer wants to use. The abstraction levels being IaaS, PaaS, Serverless (CaaS, FaaS) and finally SaaS.

Cloud providers ordinarily also offer certain affirmations to their clients in form of general maintainability, robustness, and compliance of their platform via diverse portfolios of certifications required to enter certain fields such as health or financial industries or even government hosting. There are also certain levels of service-level agreements which are provided in the basic versions of the cloud services and additional ones the cloud customers can pay for to get better agreements. Getting better agreements can also be part of initial negotiations with any given cloud provider. If the customer has any leverage, they can use to pressure the cloud provider to get a better deal.[10]

When it comes to hosting a software solution in the cloud there are several advantages which the customers are seeing as a great benefit but also several disadvantages which are often overlooked by the potential customers and the overlooked facts often lead to higher than estimated invoices and wrong ahead-of-time calculations for example during the initial return on investment analysis. The primary benefits of this are most definitely flexible billing which depends on the actual usage of the cloud services, then it is flexibility because the customers can scale up and down the cloud resources on demand based on their needs. More advantages can point towards ease of various disaster recovery tasks as a popular paradigm of most of the cloud infrastructure is provisioning the infrastructure as code resulting in mostly automated environments which can be brought up in a matter of minutes or hours instead of weeks lowering the effective Recovery Time Objective to more acceptable numbers. The disadvantages usually and primarily come in form of hidden costs, which are hard to predict costs during the initial analysis of most of the cloud providers as they are not directly implied on most of the pricing pages even though the cloud providers are trying to warn potential users that using the services in the cloud incur certain costs which may relate to data transfers or additional personnel needed to maintain the solution around a certain service. More disadvantages point more towards the need to pass certain types of responsibilities to the cloud provider so for example you are trusting the cloud provider with business critical data or maybe even confidential data which may imply huge security obligations to prevent breaches, avoid compromission of credentials and the cloud providers may not be entirely transparent about the internal security issues and practices and the customers themselves might have to often verify the security requirements are being met using several cloud security tools available online on the Internet. Another frequently discussed topic is vendor lock-in which is a term referring to a state where an organization is either unable or it is extremely hard for an organization to migrate away from a certain cloud provider to another one or back to an on-premises solution. The reasons might be diverse such as better pricing or a deal from any other cloud provider. Cloud providers can make some of their technology deliberately incompatible with other competitors, so it is hard for their customers to migrate away and thus creating vendor lock-in. Therefore, companies should create a so-called cloud exit strategy when comparing and committing to a certain solution from a chosen cloud provider, so they weigh the possible implications from migrating away from the solution in the future into the final decision.[11]

2.1 Amazon Web Services

Amazon Web Services, AWS in short, is a cloud service provider owned by, as the name suggests, the Amazon.com organization. They have been a major leader in the cloud computing industry since 2006, when the AWS was launched with the purpose of opening the infrastructure which was used to run the Amazon.com website to the general population and expand its offering with these computational services. They were amongst the first companies which pioneered the “pay as you go” model which was revolutionary for the customers as it offered them enormous flexibility and ability to scale various environments which they could not before the introduction of this platform.[6]

Among the most prominent services of the AWS ecosystem is AWS S3 and EC2, where the S3 refers to Simple Storage Service and Elastic Cloud Computing. The S3 service is an object storage service which offers massive possibilities for scaling when it comes to either writing or reading the objects from the service. S3 also offers a bunch of internal mechanisms which make the administration and management of such a storage easier for the personnel managing it. Paired together with one of the best pricings on the market for storage and various automations for making the pricing lower itself over time for unused usage makes the S3 a very solid choice for storing data in the cloud. When used with the EC2 service it can become a powerful combination of highly scalable storage and a giant computing platform with immense elastic possibilities. The EC2 platform itself grew from a simple virtual server provisioning platform into a complex platform for provisioning not only virtual servers but also related services as networking components, disks and block storages and scaling and auto scaling of the provisioned resources according to the customers' needs and demands with support for various compliance and governance requirements. The EC2 platform also became an enabling component for various services inside AWS such as Elastic Container Service for running containerized workloads in a proprietary container runtime hosted on a Docker platform or Elastic Kubernetes Service which is a service for running containerized workloads inside a Kubernetes cluster, and many more services which just simply use the EC2 platform as a service they are backed by and are creating the EC2 instances on behalf of the customer and also manage these EC2 Instances for them. [5]

When it comes to availability, Amazon Web Services offers thirty-one regions spread across the globe. Each region consists of multiple availability zones which translate into different physical zones in relative proximity, usually within a few kilometers in range. There are also over four hundred edge locations referring to the closest one the Amazon Web Services serve their customers from. All these locations, regions and availability zones are distributed among all the Earth's continents apart from Antarctica.[12]

The main reasons to choose Amazon Web Services as the main cloud service provider might include the fact, they still are the leader in the cloud computing space even after two decades of operation and the fact that their service offering is one of the most diverse ones when compared to the other cloud providers in the cloud computing market at the moment. The biggest disadvantage which comes with Amazon Web Services is their pricing structure. It is also the biggest criticism which even the current Amazon Web Services customers bring up often. The structure of the billing is truly diverse but also complex and sometimes confusing, each service offers its own and there are a bunch of caveats to be found in most of the services across the whole Amazon Web Services portfolio.

2.2 Microsoft Azure

Microsoft Azure is another cloud computing platform provided, as the name suggests, by Microsoft. It was founded in 2010, while initially announced in 2008, and provides more than two hundred cloud computing services worldwide now. The Azure platform also offers distinct levels of abstractions of its infrastructure provisioning services. Majority of this cloud platform is of course driven by the most popular Microsoft products like Windows servers, the Office software bundle and Active directory but it also offers diverse options when it comes to Linux based workloads and more open-sourced portfolio products and services.

The services which have seen the most usage is Azure DevOps, Azure Blob storage and Azure Virtual Machines. Where the Azure DevOps is one of the first services launched as part of the Azure platform. The service is primarily used for planning and better collaboration during the delivery phase of software. It is used for building, testing, and deploying using the most popular Continuous Integration and Continuous Delivery principles. The service itself includes a lot of sub-services which include Azure Repos, Azure Pipelines, Azure Boards, Azure Test Plans and Azure Artifacts where each of these services play a major part in the DevOps and Continuous Integration and Continuous Delivery pipelines.[13]

Azure Blob Storage is the object storage of the Azure platform ecosystem. It is used primarily to store massive amounts of data which is usually unstructured. The usual data stored here may include images and documents which are accessible and are served through internet browser interface on websites, we can write application logs here, we can stream video media files and we can also store data for archival purposes and backups to perform disaster recoveries. The most central piece of the whole ecosystem is for sure the Azure Virtual Machines which is a service for provisioning virtual machines with per second billing. The servers can be burstable, compute-optimized, memory-optimized, or general-purpose. It is an essential service which backs up a lot of the other services on the Azure platform.[14]

At the time of drafting this thesis, the Microsoft Azure platform offers exactly 60 regions across the world and offers Availability Zones which are the physical data center locations distributed in the relative region within an acceptable range of each other.[15]

The biggest advantage when it comes to Microsoft Azure is for sure the direct support for all Microsoft products including the Active Directories, Windows server licenses and the Office 365 software bundle. If the customer is heavily invested in the Microsoft ecosystem, then it might be a viable choice to weigh against the other cloud service providers as they are the primary distributor for this ecosystem. When it comes to disadvantages, one might consider the service offerings which are still low when compared to the leading provider Amazon Web Services.[7]

2.3 Google Cloud Platform

To join the trio of the biggest tech companies in the world, Google released an initial first version of their platform called Google Cloud Platform in 2008 and it was a preview version of the App Engine service which is offered until this day. In 2011 the App Engine service was released as a full featured version of itself, and the offering was broadened by more than one hundred products since the initial release of this first service. When compared to the two giants in the industry, Google still holds the third place falling behind after Azure which has taken off to catch Amazon Web Services, but for example still owns the original release rights for industry changing technology like the Kubernetes orchestration engine which has taken the cloud computing industry by the storm in the recent years.[16]

Mentioning that fact, the most used service is Google's Kubernetes[3] Engine which is backing up most of their platform as they are the former inventor before releasing it public in the middle of the year 2015. Google was offering this service since its release together with Red Hat as part of their OpenShift and other competitors joined them by announcing the native support 2 years later in 2017. The other services used by most of the customer base go hand in hand with the basic needs of most of the developers. The services being the original Google App Engine service, Google Compute Engine and Google Cloud Storage with Big Query service. The initially offered service App Engine is a Platform-as-a-Service cloud resource which gives the developers the advantages of Google's scaling abilities straight to their hands by enabling the developers to create the app using a cloud native process directly in the Google Cloud Platform by developing the applications remotely directly on the platform itself. The “step down” from the Platform-as-a-Service is the Google Compute Engine which is the infrastructure-as-a-Service computing service in Google Cloud Platform. This service allows you as the end user to mention the computing resource together with the necessary storage and network infrastructure which you configure. The basic premise of this service is you don’t invest in the actual hardware, space in the racks and personnel required to physically attend to these kinds of physical machines but you just must hire personnel which knows how to set it up in the UI or using scripts, Google does all the other heavy work for you.[17]

Google Cloud Platform is currently available in 35 regions spanning across all the continents but Antarctica and offers close to 200 network edge locations the users are primarily connecting to.[18]

The straight advantage is derived from the fact that Google is the original inventor and developer of the Kubernetes standard, and all the competitors are also using it to their advantage. This fact simply allows others to migrate their workloads flawlessly between the cloud providers in case of better deals or the customers can also utilize this fact for a hybrid solution and choose this cloud provider as the secondary option for their services without any significant blockers in the way. What is not so perfect when compared to the bigger of the trio is the number of locations all over the world and the sheer number of services and features included in the platform itself as the Google Cloud Platform started the last it’s still lacking some services which it still fails to offer and thus falls behind the Amazon Web Services and Microsoft Azure.[7]

3 PRICING MODELS INSIDE THE CLOUD

There are a substantial number of ways how the billing models in the cloud works. Of course the most popular amongst the organizations utilizing the cloud for the computing capabilities is the most general on-demand or “pay as you go” model, where you don’t have to predict any metrics or trends and you just essentially pay for what you really use during the usual monthly billing period where the final price is calculated using bunch of methods used for calculating the actual usage of a service using the cloud service provider internal metrics and statistics. The most used is specified in hourly rates for the specified resource and billed in per-second intervals for example for most computing resources like EC2 Instances or Virtual Machines.[4] The cloud service provider can improve this billing model for the end customers by making the pricing more granular, meaning making the scanning frequency of the state for the given service higher, resulting in more frequent checks and thus lower pricing if the service is running for example just a few hundred milliseconds. The customer then does not have to pay for the full second where the service was running only for one tenth of a millisecond. The best pricing in this category is currently offered by AWS Lambda where Amazon Web Services went out of their way to provide pricing which would bill the customer on per-millisecond basis and thus making customer’s code optimizations so much more valuable as they can really bring the final pricing of the service down significantly. Next popular pricing method is a static price for a resource monthly in the form of a subscription, for example a price per active user in a calendar month in the given service. This kind of billing is usually applied in the Software-as-a-Service world. A lot of the services also resolve to use per unit pricing where the service itself defines the given unit for proxies and content delivery networks; it can be a request, for an emailing service it can be a sent email and for storage it can be a gigabyte.[2]

Same as the original customer of the cloud service provider used to estimate and used to commit for the certain amount of hardware which had to be over provisioned to accommodate for the potential peaks and miscalculations, the cloud provider must do the same and thus the cloud provider may decide to put saving bundles into the offer portfolio together with reserved instances and resources and also a spot offering. Every one of the mentioned discounted forms of cloud computing function on an essential premise, the customer must commit to a certain minimal usage they are willing to pay for. Forms of these commitments might differ based on the customer and the cloud service provider, but they might include reserved computing time per month or year, minimum data transfer usage or minimum number of requests per month, for data storage it might be the minimum data storage used per given month. The cloud provider benefits from this information and can reserve the necessary resources for all the reserved capacity and thus provides the customer with usually an appropriate discount. Same goes for the spot offering but in the reverse way. The fact that the cloud providers must take the on-demand computing capacity leaves them with huge margins between the actual usage and the computing capacity they committed for in the form of the physical hardware, servers, disks, networking capacity and so on. Therefore cloud providers often opt in to provide this spare capacity to the customers which want to use it for a massive discount with one condition, if the cloud provider asks for the used computing capacity the customer has to give it up within the specified time frame, otherwise the customer computing power gets cut off and assigned to a customer which asked for it in the on-demand manner. These kinds of offers can be essential for lowering the overall pricing for interruptible workloads and processing jobs which can be split into parallel across different instances in a distributed manner and one job can overtake where the last one left off and so on.

4 AMAZON WEB SERVICES

The AWS platform was chosen as the primary cloud service provider for the major market lead and huge service offering where Papirfly was already utilizing some of the services for certain client implementations before the decision for a complete migration into the cloud was brought up. The shared competence of a certain number of team members of the AWS team helped enormously during the migration process as the initial learning curve was overcome quite easily. One of the main deal breakers for the Papirfly solution was the Simple Storage Service and its Intelligent tiering feature. The former BrandMaster Solution brought a lot of stale and unused data which had to be kept for compliance and contractual reasons and this feature allowed for cheap storage which is automatically controlled when the data is not used over time. This feature made storage one of the cheapest services in the overall billing from one of the potentially biggest items on the bill at the start of the initial pre-migration investigation.[6]

4.1 Description of the relative AWS services used within the environment.

The Papirfly environment consists of tens of services provided by Amazon Web Services and their usage span from a simple file storage to most advanced usages as autoscaling certain instances based on specific metrics collected from databases and combined with specific thresholds they provide a way to figure out the needed instance number at a given time. These short and concise descriptions should help any potential readers with different knowledge levels of the Amazon Web Services ecosystem to understand the mentioned interconnections within the environment and for what are the services used.

4.1.1 Virtual Private Cloud

Amazon Virtual Private Cloud is a service which gives you full control over organizing your virtual networking in your environment, you can define where you place resources in terms of subnets or availability zones, you can also configure various connectivity services which are hidden underneath this service. Papirfly uses this service for specifying its network structure using subnets for each availability zone in duplication for public and private subnets connected via NAT Gateway and connected to the internet using the Internet Gateway. There are also a bunch of endpoints specified for the Amazon Web Services to route to certain services internally instead of going over the public network. The internal routing is specified in the routing tables and the networks are separated by network access control lists and security groups.[20]

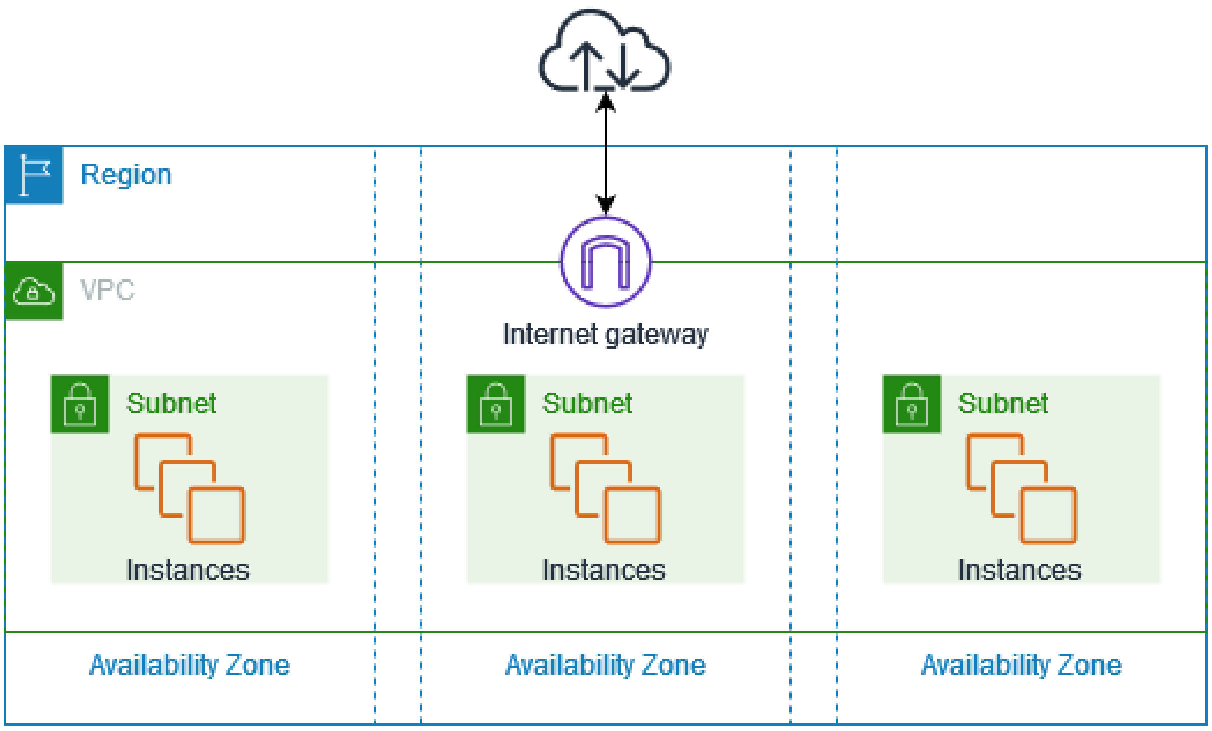

Figure 3 – Simple VPC Architecture [20]

4.1.2 Web Application Firewall

AWS WAF is, as the name suggests, a generic Web Application Firewall which is enriched by a lot of managed functionality by AWS. There are a bunch of pre-made and managed rule sets by AWS which can protect the solution from the most common types of vulnerabilities and prevent exploits from even entering the environment. There is also managed monitoring which even exports the logs into the specified consumer service and the customer can analyze the logs further from there or connect it to some of the analysis services available inside AWS.[21]

4.1.3 Lambda

AWS Lambda is a serverless computing service which lets you as the customer run any type of code without having to provision a single server and you only pay for the actual runtime of each function in millisecond intervals. Lambda is useful for processing data at scale and in event driven environments as the Lambda environment itself can react upon events sent from over 200 services inside the AWS ecosystem. Lambda can also be used for processing on edge and this sub offering of the Lambda service is called Lambda@Edge where the lambdas are launched on request in the CloudFront service and can be used to mutate HTTP requests and responses.[22]

4.1.4 DynamoDB

DynamoDB is a NoSQL database service which is scalable to even petabyte scales while the AWS manages operation tasks like encryption, recovery and backups and there are also free data modeling tools using the AWS best practices. The DynamoDB service is often used as a key value service in the Papirfly environment as the service itself is fast to spin up and provides awesome scaling opportunities for read access for the processing at edge where the domain configurations and redirects are read. Realtime changes to the database can also be streamed into other AWS services for further processing or analysis using the built-in stream feature.[23]

4.1.5 CloudFormation

CloudFormation is the AWS in-house service for managing your infrastructure as code which is a popular paradigm among the modern DevOps community as it ensures the environments are re-creatable using these templates which can also be versioned and carefully reviewed in a team setting. CloudFormation uses YAML for specifying the AWS resources and their configurations and offers enormous possibilities for extensions using the AWS Lambda service where developers all around the world can build their own extensions and additional logic to enrich the templating possibilities of CloudFormation.[24]

4.1.6 Elastic Cloud Computing

The EC2 service is the heart of the whole AWS ecosystem as it backs up most of the computation intensive tasks inside AWS by provisioning virtual instances of servers with many sizes and parameters which can include diverse selection of vCPU, GPU, RAM and other specialized hardware and its architectures to suite the customer’s requirements and demands. The EC2 service hides a lot of sub-services as it used to be also a load balancing service and a networking service and some of the functionality has not been moved out of it, at least not yet. There are various possibilities to set up networking components like Elastic IP addresses which can be assigned to the EC2 Instances. There is also a possibility to provide Elastic Load Balancers and Autoscaling Groups to group EC2 Instance for auto scaling purposes under rule sets which act on different events within the AWS system.[27]

4.1.7 Elastic Load Balancing

The Elastic Load Balancer is part of the EC2 service in the AWS Console UI and is further down split into multiple types allowing the end user to choose the right variant for their use-case. The service consists of an Application[25] and Network Load Balancer[26] and a specialized Gateway Load Balancer, every one of these types serves a special purpose. The Application Load Balancer is a standard Layer 7 application communicating on a HTTP/HTTPS protocol and supports gRPC and WebSockets on the HTTP 1.1 / 2. It also provides various built-in features like health checks of the end targets, content-based routing, various load balancing algorithms and provides sticky sessions functionality. It is also possible to terminate the TLS using the load balancer and it can also have AWS WAF attached to it. The Network Load Balancer is a typical Layer 4 TCP/UDP load balancer; it provides connections on a lower level when compared to an Application Load Balancer and thus also offers limited functionality. There is support for TLS termination, health checks and Zone Isolation and long-lived TCP connections. The last Gateway Load Balancer type is a specialized load balancer which operates on the third layer of the OSI model, and it listens for all IP packets across all ports and forwards traffic to the specified target group specified by the internal rules and thus it's usually used to manage virtual appliances such as firewalls, intrusion detection and prevention systems and deep packet inspection systems.

4.1.8 Elastic Block Storage

Elastic Block Storage is the final service which is still part of the EC2 service inside the AWS Console UI. It serves as a storage volume service for the EC2 ecosystem. The service itself provides raw and unformatted block devices which translate to a storage volume when attached to an EC2 Instance and the service manages the lifetime of these devices independently of the EC2 lifetime, therefore they can be for example dismounted from one instance and attached to another and used for various purposes like that. The recommended use is to use EBS for data which is to be accessible in a quick manner and needs to be persisted for an extended period. The service offers a bunch of classes the customer can choose and decide what kind of power they need for these volumes based on several classes which rely on the access speeds or throughputs and access patterns which are then projected onto their pricing.[28]

4.1.9 Elastic File System

Elastic File System is a service which provides a serverless elastic file system storage which can share data without any need for provisioning the actual capacity and additional parameters. It is exposed via Network File System in version 4 so the applications and tools can use this file system as standard system mounts. The file system itself scales based on the data which arrives to the system, on demand to even petabyte scales. It also automatically shrinks when you delete data so you as the end customer pay only for what you use in terms of storage and performance needed to store or delete the given data.[29]

4.1.10 FSx For Windows

FSx is a service which offers a bunch of file servers backed by various implementations in the backend. The FSx for Windows provides a Microsoft Windows file server backend by native Windows file system. It is usually used to migrate legacy applications into the cloud in the lift and shift manner. This file server works for the Windows workloads which are dependent on the Server Message Block industry standard.[30]

4.1.11 Simple Storage Service

Simple Storage Service, shortly called S3, is an industry-leading object storage service offering huge scalability and performance possibilities. It offers 99.9999999% of data durability and offers up to 3500 PUT/COPY/POST/DELETE operations or 5500 operations per second on a partitioned prefix in each bucket. The service offers a truly diverse selection of optimization features such as different tiers based on the access patterns or just solely based on the pricing chosen by the customer if they choose to operate in a cheap but less reliable variant of the S3 storage. S3 also offers various archival features which are provided through a sub service called S3 Glacier which has a few special tiers of storage where the customer uploads the given data which are stored in an exceptionally durable cold storage for a much lower price, the only condition being no direct access on demand. The customer can ask for a so-called data retrieval and AWS provides the given data within specified intervals which are dependent on the type of storage chosen. The archive tier is usually retrieved within 12-24 hours, the deep archive can take up to two days. S3 also offers a bunch of features such as transition cycles to different tiers, object replication and can of course integrate with the AWS Lambda service for further processing of the objects based on different events which can happen inside S3, for example an object upload.[31]

4.1.12 CloudFront

The CloudFront service is a content delivery network which consists of globally distributed proxy servers managed by AWS. This network of distributed proxies can be protected by the internal AWS WAF service and is by default provided by the AWS DDoS protection. The service is distributed in more than ninety cities within more than forty-eight countries and has more than 450 Points of Presence which are the endpoints the users' access within their web browser when they go to a website running behind the CloudFront service. As the usual nature of any CDN is to cache data, the CloudFront service offers rich possibilities to cache data on the edge locations with options to additional computing on edge either by lambdas or special CloudFront Functions which are globally distributed JavaScript files which are ran on every request on the edge location itself. The service has additional features related to video on demand streaming, it is also integrated with the internal AWS services for advanced reports, metrics, and logs.[32]

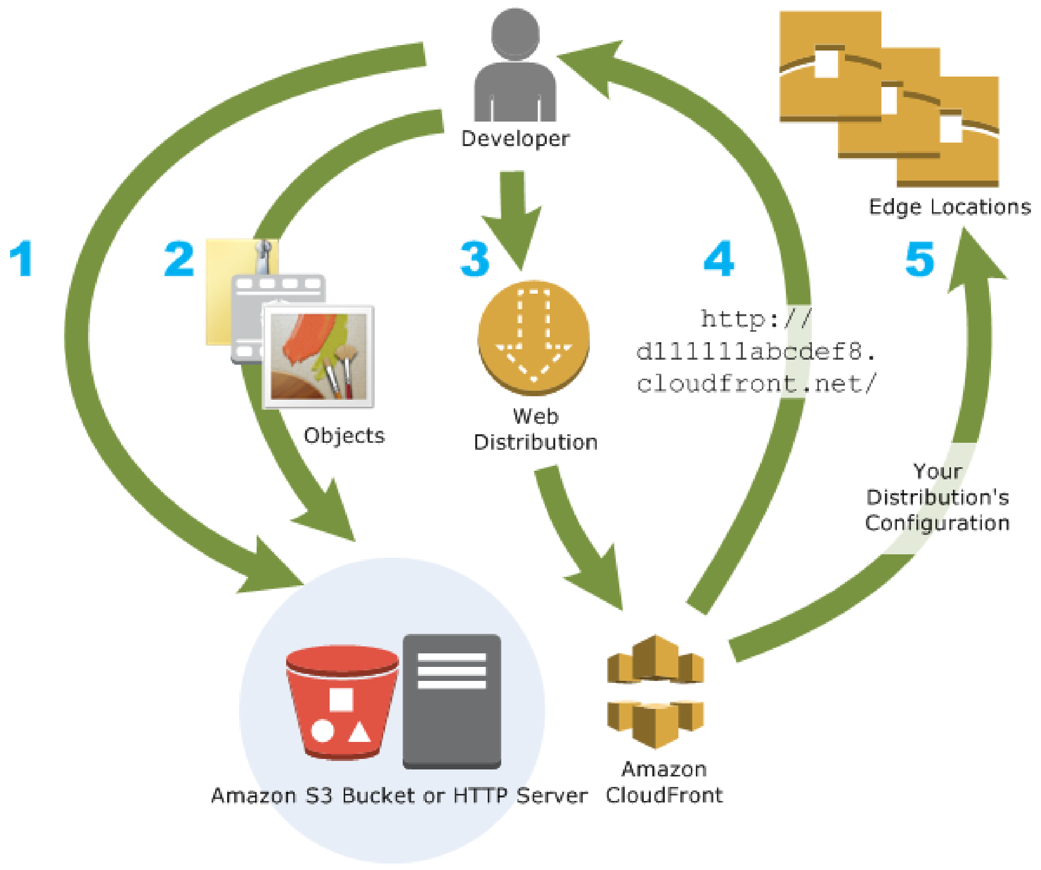
Developer
Edge Locations
1
2
3
4
5
Objects
Web Distribution
http://d111111abcdef8.cloudfront.net/
Your Distribution's Configuration
Amazon S3 Bucket or HTTP Server
Amazon CloudFront

Figure 4 – CloudFront traffic flows [32]

4.1.13 API Gateway

API Gateway is a service for managing access to APIs. It offers a managed experience for creating and publishing APIs, so it acts as the front door into applications. The gateway is secured by various authorizers which can be managed by AWS, or the developer can create custom ones with the help of AWS Lambda. The service also provides API documentation in the form of the OpenAPI Swagger style documentation which has become an industry standard over the years it exists. The APIs can be created as HTTP RESTful APIs, or the API can also be exposed using WebSockets. Additional features for this service include throttling, API versioning, built-in monitoring, CORS support and this service is also part of the AWS Free Tier program so the first 1 million of API calls to the API Gateway are free every month for the first 12 months after the AWS Account is created.[33]

4.1.14 Route 53

Route53 is essentially a highly available and scalable Domain Name System service. The user can register a domain inside this service and then configure a DNS based routing for the given domain and additionally also configure health checks where the user can be potentially routed towards a different DNS defined in a fallback fashion. The service also provides several types of routing, for example routing based on a weight which is specified using percentages and the given percentage of users is then routed to the given DNS records. The next example of this can be geolocation-based routing or latency-based routing.[34]

4.1.15 Elastic Kubernetes Service

EKS is one of the managed services inside AWS and as the name suggests, it is a service which provides the customer with a managed control plane for the Kubernetes cluster. AWS provides several ways on how to host your clusters one of them being in the cloud inside AWS and back the cluster by either EC2 Instances which is the most similar when compared to hosting an on-premises cluster or it can be backed by the serverless Fargate platform. The other options include also running the cluster on your own self-managed virtual machines or completely running the installation the customer's on-premises hardware, where it is also possible to create a hybrid solution with some of the nodes running inside the cloud environment and some of them running on-premises. The service integrates flawlessly into multiple internal AWS services like the Elastic Load Balancers, the EC2 platform and the logging and authentication tools provided by AWS.[35]

4.1.16 Elastic Container Service

Elastic Container Service is like the EKS, but it is a completely proprietary and closed and thus also completely managed container management service which allows the end user to configure and run containerized workloads also on either EC2 Instances or using the serverless Fargate platform which takes care of the server provisioning. The service is integrated into the AWS platform with support for logging, monitoring, and other features available inside the service. AWS also transparently works on this service with a publicly exposed Roadmap.[36]

4.1.17 Simple Queue Service

SQS is a service which offers a managed solution for queueing by providing secure, durable, and highly available queues to provide applications with a place to store and send messages to decouple themselves in a distributed environment. The service offers standard queues which act as at least once delivery queues with best effort ordering which means the messages might be delivered more than once and the ordering might not be preserved but the queue gets unlimited throughput. On the other hand, the other queue type, the FIFO queue provides high throughput with support up to 3000 calls per second with the batching of messages enabled, this queue acts as the standard messaging queue, so the message is delivered once, and the order of the messages is preserved.[38]

4.1.18 Simple Notification Service

SNS is a fully managed Pub/Sub service which exposes application-to-application and application-to-person notifications via integrating with a queuing solution or with SMS, push notification and email providers. The service itself supports creating topics to which the notifications are published and can be filtered or fanned out after that. There is also support for a dead-letter queue if the receiver is unavailable for a long time, the messages are kept there for further analysis or potential reprocessing after a specified time period.[37]

4.1.19 EventBridge

EventBridge is a serverless service which connects applications or infrastructure components using events distributed via the central bridge. Applications can use this bridge to listen to certain events and act based on them. There are various settings available inside the service which enable setting of granular routings between the services which may include many to many routings or just point to point routings where only two applications send events between each other. There are also options for archiving events and replaying them and you can also use EventBridge Pipes to further process the events inside the service before delivering the events.[39]

4.1.20 Simple Email Service

AWS SES is a service which enables the users to send emails in the cloud in a managed way. The service takes care of most of the pain points of running a SMTP cluster on premise or makes it easier to manage certain functionalities. The service itself provides security with SPF, DMARC or DKIM, and supports sending emails using either your own IP addresses, dedicated or shared IP addresses provided by AWS. There is also support for various marketing-oriented functionalities like engagement reports on the emails sent through the Simple Email Service. The only metrics the customer of this service must be aware of are the bounce and complaint metrics as they measure the quality of the emails being sent. There are certain percentage levels set by AWS and once they are surpassed, the account is blocked from sending emails out of their infrastructure.[40]

4.1.21 Amazon Certificate Manager

The Certificate Manager service provides the functionality to provision, manage, and deploy SSL or TLS certificates. The certificates can be issued by public or private authorities and can be further attached to the internal AWS services. The main purpose of the service is to issue the certificates once and then forget about them, these auto renewed certificates must be validated through specialized DNS records on the domain the user wants to issue the certificate for. The service further allows the customer to import certificates if the compliance reasons require it. The Certificate manager, however, cannot auto renew these certificates and they must be imported manually again when they are close to their expiration. The service monitors all certificates and notifies customers about the event of expiration 30 days prior to the certificate expiration, and it issues a new warning which notifies the customer each day within this 30 days' time frame.[41]

4.1.22 Amazon Kinesis

The Kinesis service is quite like the Simple Queue Service which is also available inside the AWS ecosystem. This service, however, is a bit different in how it operates internally. The baseline of the service is that it represents a distributed and highly durable and available linked list in the cloud. The main differences when compared to the SQS are - the messages sent into the Kinesis are not deleted upon consumption, and it can also have as many consumers as we want, simply because of the nature of this data structure. The structure is append-only and the messages in the list simply age out according to a set time to live constant, which is 24 hours by default. However, the customer must manage the capacity provisioning of the Kinesis service which is split into so-called shards which have to be monitored for usage and scaled accordingly. Because of this the service can be quite cheaper when taking just the volume of sent messages into account.[42]

4.1.23 Amazon Firehose

Firehose is an extension service for the Kinesis service itself. It can take in the data streamed in the Kinesis stream and transform it into known formats required by various external tools and other AWS services like OpenSearch, Redshift or S3 files in the given formats. All these operations are backed mostly by AWS Lambda so the customer does not have to provide any computation resources and if needed custom transformations can be built using the AWS Lambda platform using any supported programming language.[43]

4.1.24 Relational Database Service

AWS RDS is a service for provisioning Database Engines. It is fully managed meaning the user does not have access to the underlying system and the only interface for the customer is either the AWS Console UI or the connection to the database engine itself. The service then automates and manages all the mandatory operations like security patches, data durability and redundancy, performance and scalability, monitoring and alerting together with backups and their recovery. The service supports most of the popular relational database engines like MySQL, MariaDB, PostgreSQL, Oracle and SQL Server, it also provides compatibility for the Aurora distributions built by AWS for MySQL and PostgreSQL which integrate better towards the AWS services and sometimes offer even native support for some of the services and as these distributions are built for the AWS infrastructure, they might provide better scaling possibilities and increase control over certain aspects of the life cycles of the database engines.[44]

4.1.25 Amazon OpenSearch

AWS OpenSearch Service is a forked continuation of the Elasticsearch and Kibana managed by AWS teams. This open-sourced version was created as a reaction to the original

Elasticsearch going under the Elastic License or Server-Side Public License which does not offer the same freedom to most developers. Because of this fact AWS decided to maintain and develop a fork of the software under the permissive Apache License in version 2.0. The service offers the same set of APIs and because the service is developed and maintained by AWS it integrates very well with the whole platform and other AWS services. There is also support for several security options and flawless green/blue upgrade deployments which can also be done in an automated way. [45]

4.1.26 Amazon Athena

Athena is an interactive SQL service which can query over twenty-five data stores which can be AWS native or external. The service scales into Petabyte sizes and supports various open-table and file formats you can scale in a simple and flexible way. As a customer of the service, you pay only for whatever processing time is used to query the data and for the concrete amounts of data being scanned by the queries. Therefore, it is important for the data store architects to properly partition the tables according to certain parameters like date time timestamps or any other denominators. The most useful feature of the service is the ability to query data stored inside the S3 which is usually the primary storage for application logs in an affordable way.[46]

4.1.27 Amazon Systems Manager

AWS Systems Manager is an operational hub for any AWS environment. It is one of the biggest services inside AWS as it consists of 13+ sub-services and the AWS teams are still expanding it to provide increased functionality. Some of the features can manage the state, inventory, and certain schedules of the AWS resources within the AWS environment. It can apply matches and schedule maintenance windows for the whole ecosystem of various resources inside AWS. It can also automate certain tasks and run commands on whole groups of resources in a managed way. There are also ways of automating actions as a reaction to certain types of reported incidents so it can be the first-row support for simple cases of restarts which can be done in an automated way before contacting a human after hours agent to check on the given service and make further actions to resolve the matter. As this service's nature is given, it also can access the EC2 Instances using its own proprietary protocol which is handled within the AWS permission system and can bypass all the security system restrictions on the EC2 instances. If the user accounts are protected well by strong password policies and multifactor authentication methods, it allows for a secure way of connecting to the instances via various protocols like SSH and RDP without needing to distribute and issue any VPN certificates or additional user accounts and everything can be managed inside the AWS IAM service.[47]

4.1.28 ElastiCache

The ElastiCache service is a fully managed service for managing in-memory caching services which can accelerate the response times and performance of databases and applications. The caching services can also act as a primary datastore for short-lived data like sessions shared between different auto scaled instances of applications, streaming or data analysis. The service itself offers Redis and Memcached as the underlying engines which can also be scaled in a managed way, they are also monitored and back-end up accordingly.[48]

4.1.29 Amazon Transcribe

Transcribe service is the automatic speech recognition service for the AWS ecosystem. It takes in any audio form of a speech and transforms it into a text form. It can detect multiple languages and partition the speech of individual speakers in the audio file. The service itself supports two modes, one of them being batch processing which is an on-demand way of processing the speech data, the other way is real-time streaming processing which is ideal for real-time conversations. The service itself is ideal for providing subtitles generation or generation of transcriptions of conferences and so on. The service itself can also detect certain analytic metrics like speech interruptions, talk speed and even the speaker sentiment.[49]

4.1.30 CloudTrail

The CloudTrail service monitors the API activity of the given AWS account and tracks all the actions in the form of trail events. The events identify who did what and when with addition of additional metadata relevant inside the AWS ecosystem. The service itself has support for generating audit reports which can validate the authenticity of the stored and monitored records. The records stored inside the trail can be further analyzed and monitored for anomalous activity and even operation issues, you can for example see if the system is throwing out an unusual number of forbidden accesses to certain services which have been changed and so on.[50]

4.1.31 CloudWatch

AWS CloudWatch is a service for collecting and visualizing application logs and metrics running inside the AWS ecosystem. The service can also be used as a generic monitoring tool and send escalation messages to an on-call personnel rotation service as OpsGenie. Specific metrics can be also monitored and act upon for the purpose of auto scaling or the customers can also set up metric alerts for monitoring the bounce and complaint rates inside the AWS Simple Email Service and automatically block the email sending once it gets over a certain level which might incur potential problems with AWS policies which might result in blocking the email distribution for the account.[51]

4.1.32 Transfer Family

The Transfer Family services enable the AWS customers to host their own file servers accessible though SFTP, FTPS or FTP protocols in a completely managed way. The service itself is then backed up either by S3 or EFS storage and the user base can be managed by using a third-party data store through custom authorizers made using AWS Lambda or by using a managed datastore directly in the Transfer Family service. This makes it ideal for migrating already existing servers without the need to manually manage the underlying infrastructure and can be potentially cost effective on the right scale.[53]

4.1.33 Storage Gateway

AWS Storage Gateway is an application and a service which provides an abstraction of the AWS native storages and makes it so the storages can be used in more conventional ways like NFS/SMB shares, iSCSI devices or iSCSI VTL devices which are usually mounted on the target operation systems. With this ability of acting as these devices the application data stores can be migrated without migrating the application itself which can often be beneficial to reduce complexity of the migration. The mounted devices can be substituted with these abstractions and the application can be gradually migrated over into the cloud.[52]

4.1.34 Identity and Access Management

AWS IAM service is the central part of the whole AWS ecosystem as it is the service which every user of the AWS will encounter. The service specifies all the users, their security credentials and other security precautions like multi-factor authentication, key management and rotation, certificate management and so on. The service also specifies policies, roles and user groups which say which user has access to what services. The service also allows for so-called temporary security credentials which allow the users to use separate roles. The roles themselves can be part of different accounts and thus provide the given user with the ability to jump across completely distinct accounts connected by a trust relationship between the role and the source account. These relationships are entirely controlled via the IAM service and can thus be managed in a centralized way. One of the common ways to separate different environments inside AWS is creating so-called multi-account environment where the various levels of the infrastructure are separated by the account constructs and certain users are given specific permissions on the given accounts so they can fulfill their duties on a very grain-defined sets of services which can span across many AWS accounts.[54]

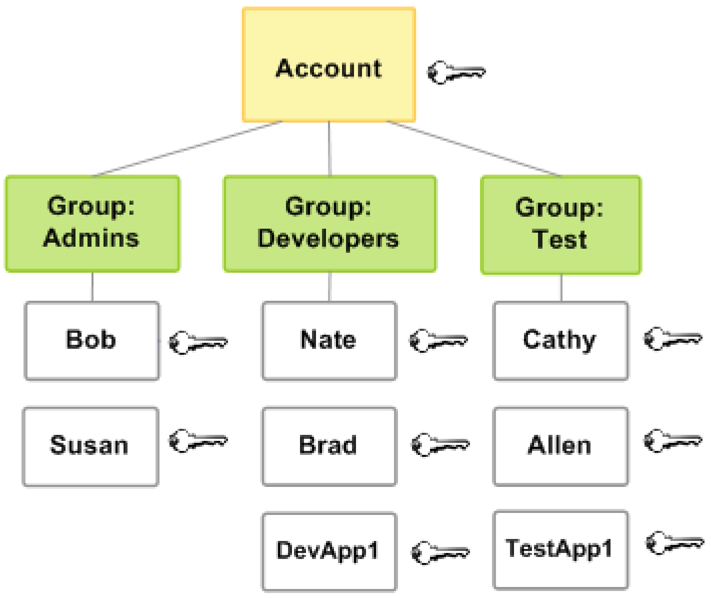

Figure 5 – IAM Structure [54]

II. ANALYSIS

5 AFTER MIGRATION PROCESSES

The Papirfly AS organization, formerly known as BrandMaster AS, merged with the former Papirfly AS which stood as a standalone company before the merger. The former BrandMaster AS company initiated a business case of moving its entire operation into the cloud and chose Amazon Web Services as the primary cloud service provider to target during this migration. The migration was successfully fully completed in the first quarter of 2022 and the company has been running in the Amazon Web Services cloud environment ever since and has shut down its operation in the former Norwegian Blix solution server hosting, where all the unused data and configurations were moved onto tape backups for archival purposes.

Even though the migration into the cloud was an overall success, the environment the applications were running in was not entirely optimized due to the nature of the time plan and tight schedule set by the doubled costs which were implied by running a duplicated setup on-premises and inside the cloud environment. Major part of the services was migrated in a lift and shift fashion which by itself can incur additional spendings in the cloud as the most cost-effective way is of course the serverless or more cloud-native way of hosting the services. Furthermore, the solution already migrated into the cloud was effectively duplicated because the setup was primarily done inside a stage environment where it was thoroughly tested and put together as a whole and then duplicated into the production environment from where it was rolled out and exposed towards the customers. All these facts imply there is going to have to be a development cycle focused on bringing the costs in the cloud down and implementing certain measures and processes in place which will further future proof the cost control inside the whole cloud environment.

The whole after migration situation was even more complicated by the merger of the two entities where all the infrastructure was consolidated under one umbrella and even more optimization cases were brought up because of that because it was evident even from the initial analysis that some parts of the infrastructure and services were not taken care of for an exceptionally long time period.

6 TOP-DOWN VIEW OF THE AWS ENVIRONMENT

The cloud environment running inside Amazon Web Services consists of a relatively high amount of accounts, where an account is a container within the AWS itself which gets an identification number and all the services are run in this given container, users can login directly into the container and other accounts can directly or indirectly also interact with this account container if there is a trust relationship established between the two accounts. The Papirfly environment consists of fifteen primary accounts and then there are dynamically created accounts which are called developer sandboxes. Each developer can request an account to have a place to experiment and try out different services during the development phase. The primary accounts used by the organization contain the current staging and production accounts, together with utility accounts which hold the supportive infrastructure like logging OpenSearch clusters, backups and continuous integration and continuous deployment workers.

The environment was initially setup to be built of 5 essential accounts which would hold the main parts of infrastructure together but it became evident that this initial architecture proposition will not be sustainable and the accounts would have to be kept either in the old structure as it will not be possible to migrate the application to the new accounts in the given timeframe and it will become increasingly more and more complex to separate the applications from each other on the IAM level inside AWS. Even the access setup for each developer has become quite a grueling task which involves a lot of trial and error during the initial setups and quite a lot of conditions to properly define the granular access for each developer group.

Further down the road after the migration has finished, a decision was made to follow the multi account architecture for splitting up the environment into the account container units as it is easier to manage on larger scales. The multi account setup implicates more secure environment by design because the user groups within the AWS eco system must be explicitly allowed to be let inside the given account by the IAM policies and furthermore a trust relationship must be established between the different accounts so the users can access resources in the relative accounts with users centralized in other account.

6.1 Identity & Access Management Account

The nature of the setup requires the environment to have an entry point account from where you access the rest of the environment in a secure and reliable way. In AWS, the IAM and

STS service secures this functionality. The IAM, short for Identity & Access Management, is a service which holds the specification for the users, user groups, roles and policies which are enforced in the entire AWS ecosystem. The STS is a natural part of the IAM system as it provides security tokens which are time-based credentials which can be only issued with the adequate level of privileges for the given account the user is requesting the credentials from. The benefit of using this setup is that the accounts are logically separated this way and in order to get into an account, the user must go through a series of initial events like setting up a multifactor authentication method associated with their account and this setting can be properly enforced as it can be a requirement for the role assuming functionality which acts as the functionality for accessing the other accounts from central IAM account.

6.1.1 Role Assuming

Assuming a role in AWS means getting a set of temporary credentials which give the user access to a given role which further specifies the access to different resources in AWS. The role can be assumed either within the same or different account and that's how users and services can communicate with resources which span across multiple AWS accounts.[19]

To assume a role, the user must have an IAM permission to assume the given role. However, that is not the only thing which has to be set up, additional trust relationships must be set up between the given accounts to make the role assumption work. The trust relationship is set on the AWS role in the target account, where the source account has to be specified in the trust relationship, specifically there can be a selected user, or a selected list of users specified in this trust relationship or there can be a whole account specified where all the users from that account can assume the given role after this kind of trust relationship is established.

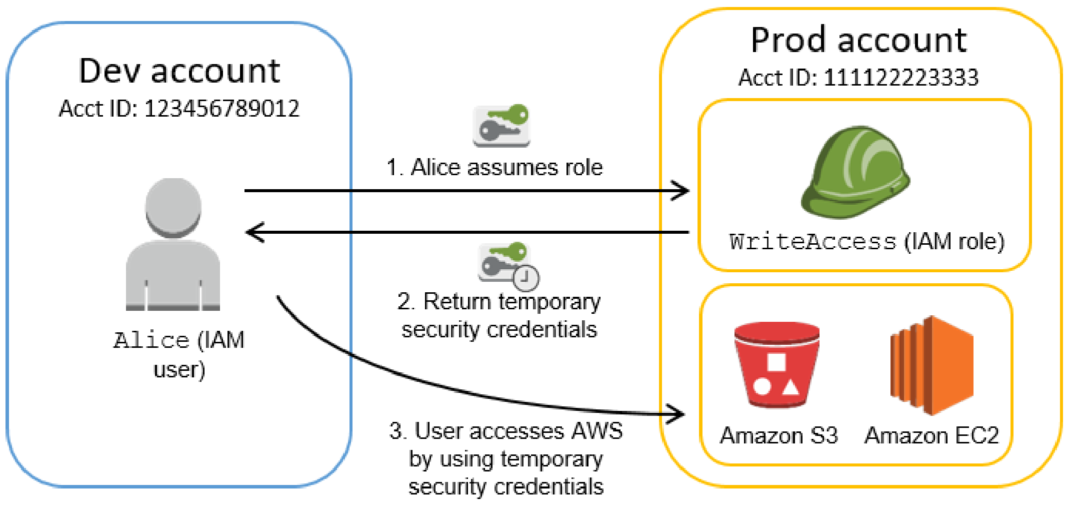

Figure 6 – Assume role flow [54]

6.2 Application Accounts

After the user properly sets up his IAM identity and its multi factor authentication, access to the other accounts open for that given user. Application accounts are the primary accounts which most of the developers interact with during their day-to-day operation and development. The Primary accounts are the Staging and the Production account, with the Development account being a dynamic component.

The development account in this environment is an ephemeral component which is brought up only if a use-case specifically requires it and it was decided to abandon the concept of a third duplicated environment for the development purposes. Instead, the concept of developer sandboxes is utilized in day-to-day work. This concept provides each developer with their own AWS account which they can access upon setting up their IAM identity correctly, same as when accessing the standard application and utility accounts. It provides the developers with the possibility of experimenting within their own and completely separated container within AWS, so every service can be tested in a live fashion and does not have to be emulated locally, with LocalStack for example. This alone pretty much eliminates the need for a separate development environment, and it led the Papirfly environment to significant savings on infrastructure. Therefore, the Development account is a container within the AWS which is most often used for spinning up the whole infrastructure for testing purposes of the Infrastructure Team and for Disaster Recovery purposes.

The Staging and Production accounts are 1:1 duplicate with minor changes done inside both accounts for optimization and customer modification purposes. The accounts are the copies of the formal on-prem environments, so every application and server run within the boundaries of these accounts. It is worth mentioning these accounts are a major part of the entire bill for the entire AWS environment as basically the whole customer facing solution is running inside these accounts. Everything from the Kubernetes cluster and storage to the Internet facing CDNs are hosted in these accounts.

There are also two new applications, Planner and Reporting, accounts which are completely separated application accounts for two brand new application introduced into the Papirfly solution and as they are being actively developed now the need for a separation was met by creating these completely separated accounts where the application run the respective development teams are working on the given applications.

The other accounts in the primary group are mostly serving as support accounts for the main accounts in the form of workers for the CI/CD pipelines and a place for storing the backups and logs from all the accounts in a centralized way.

Every other account in the environment is kept mostly for legacy applications or special projects. The former Papirfly and BrandMaster accounts continue to host components which are not ready to be integrated to the new consolidated infrastructure which the company use ever since the merger between the two entities was completed or they are being used for special projects for specific customers in case of the former Madeo account, which was the initial Czech entity until it was bought by the BrandMaster company in 2017. The other, former Tactic and former Resolut, accounts are also two tools which were acquired by the BrandMaster company in separate years and where the Resolut application is in a maintenance mode, so it is not going to be migrated into the ecosystem and the Tactic application is in process of moving over to the internal infrastructure now.

6.3 Top-down view of the Infrastructure inside the Production account

When looked at from the incoming traffic point of view, the first thing in front of the whole solution is the Route53 DNS which CNAME the given distribution for the assigned domain which is usually referred to by the CNAME record by the customer’s domain or one of the internal customer subdomains of the papirfly.com or brandmaster.com domains. As it is shown on the Figure 7 – Simplified routing, The next natural component would be the CloudFront CDN which Amazon Web Services offer; however, this entry point is protected by the Web Application Firewall service inside AWS so that is the next stop on the way into the solution. The traffic is evaluated against various internally managed and custom-made rules related to exploits and most common vulnerabilities as well as number of both specific rules and blackholes for badly reputed IP addresses. Only after these evaluations can the traffic hit the actual CloudFront edge location from where the traffic is routed to the next layer of infrastructure. For routing purposes there is an additional layer of processing before the requests reach the mentioned infrastructure, which is backed by the AWS Lambda platform, namely AWS at Edge which provide a reliable and scalable way of additional processing of the requests coming into the solution with the opportunity to cache the results of the given processing for better performance. The Processed request can then essentially continue to one of two locations, either an S3 Bucket to serve static content or to one of the load balancers for further routing into the depths of the solution system.

If the request is routed to an S3 bucket, then the flow of the data is quite straightforward as the only thing there is left to do is to access the S3 bucket on the requested path and determine if the CloudFront origin identity has the necessary right to access the object or if the object even exists within the bucket the request is routed to. Additionally, when the data is flowing back as a response to the given request, the data can be further processed and cached on the edge locations, the end user sent the request to, for even more improved performance on the consequential calls to the edge locations for the same content. The content is cached on the edge locations using the request URL as the cache key for 15 minutes and the cache can be bypassed by a query parameter called cacheVersion, which can be specified within the applications to immediately request fresh content from the origins which are otherwise cached on the CloudFront edge locations.

The Figure 7 – Simplified routing further shows the second route, which takes the traffic into one of the internet facing load balancers which are further protected from the outside world using the AWS WAF so only the traffic coming from the CloudFront CDNs can get through. The load balancers can be split into two distinct ones based on the traffic destination. The first one takes the traffic into the AWS EC2 service where most of the legacy components are being hosted after the lift and shift migration from the on-premises solution. These are mainly Windows Instances running the old legacy Java or Delphi code and one of the servers is also a proxy to the Oracle Database as a lot of the core functionality of the Papirfly solution is still dependent on codebase written using Oracle's PL/SQL stored procedures which are invoked through a tomcat server standing in front of the Oracle Database or from within the actual applications. The second primary load balancer stands in front of the Kubernetes cluster which serves as the heart of the whole solution as most of the applications were migrated into a containerized form and made Kubernetes friendly, so naturally the Kubernetes cluster is the preferred way of deploying the new application if the development team does not want to opt in for the completely serverless variants of the hosting environments. The load balancer is controlled from within the Kubernetes cluster using the AWS Load Balancer Controller which builds the load balancer routes with the specifications of the Kubernetes Ingress rules. The rules specify a route and where it should lead within the cluster. It usually points to a specific Kubernetes Service object which routes the traffic further down the Kubernetes object chain until it gets to the lowest Kubernetes unit, a pod, where the request is processed using the application web server running within the pod's container. The service can then call other services within the AWS environment, call a database, or return the result straight away if it is cached in any form.

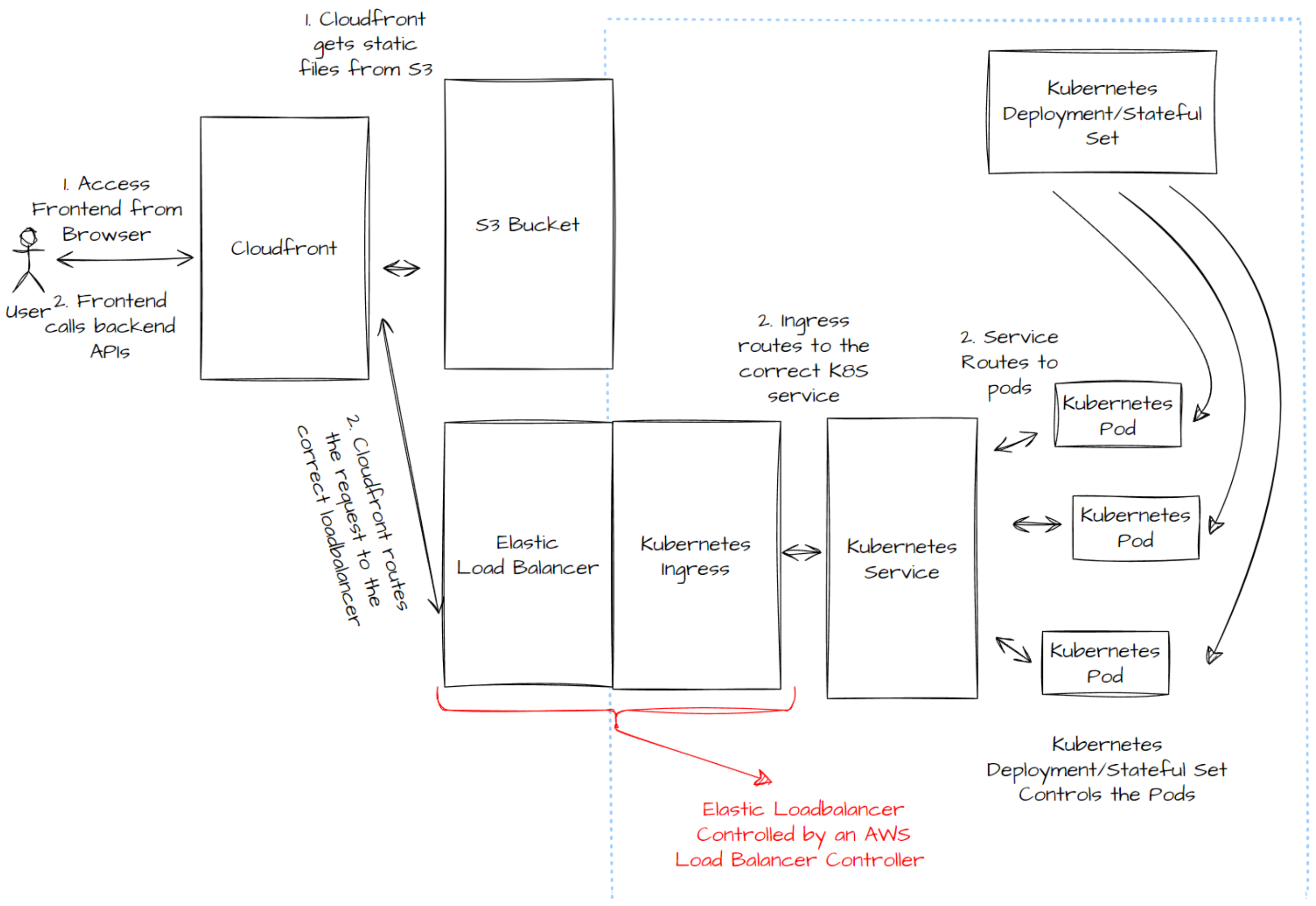
1. Cloudfront gets static files from S3
Kubernetes Deployment/Stateful Set
1. Access Frontend from Browser
User
2. Frontend calls backend APIs
Cloudfront
S3 Bucket
2. Ingress routes to the correct K8S service
2. Service Routes to pods
2. Cloudfront routes the request to the correct loadbalancer
Elastic Load Balancer
Kubernetes Ingress
Kubernetes Service
Kubernetes Pod
Kubernetes Pod
Kubernetes Pod
Kubernetes Deployment/Stateful Set Controls the Pods
Elastic Loadbalancer Controlled by an AWS Load Balancer Controller

Figure 7 – Simplified routing

7 APPLICATIONS RUNNING INSIDE THE AWS ENVIRONMENT

From the architectural point of view the Papirfly solution consists of three general setups of applications built at different points in time. There are legacy applications which are in a maintenance and sunset mode, containerized applications which are being actively developed and maintained depending on the current use-case and duplicity identification during the unification process when the two entities merged and a completely new serverless applications which are the latest generation of applications running inside the environment. Only a few selected development teams started to experiment with certain services during their development process, so these kinds of applications take only an exceedingly small percentage of the overall solution.

7.1 Legacy applications

The legacy applications can be split into four categories. Windows Delphi applications, Java Legacy application and the ORDS servers, which is the short form for Oracle Rest Data Services, and a third-party installation of a document and template editor called CHILI publisher.

The Delphi services are mostly built in a monolithic way so there is one large instance of hundreds of SOAP API endpoints which take care of various functionality from simple CRUD operations to complex image manipulation actions which the former solution was built upon. This solution is supported by Delphi encode servers which take care of the processing of all the image and video manipulations. These servers serve as workers which simply take the requests coming in from a database. They are clustered in a primary to secondary fashion and they are also being auto scaled depending on the monitoring of the length of the queue which is backed by the central Oracle Database. This solution all together is called DAM 1.0 Transcoding as it takes care of the customer requested transcoding in one of the old BrandMaster products. The whole solution is also supported by a Sync and a Utility server which provide a secure way of interacting with the backends through a user interface built as Desktop applications stored on the servers directly. These applications are accessible through the RDP protocol and thus they provide a secure way of interacting with the old system as it is quite a long way, protected by several security measures, until the user gets to the server and the user itself must have several privileges confirmed by an Administrator within the system.

The old legacy application is a remnant of the past within the old BrandMaster application ecosystem as this server used to serve an application which was entirely Flash based, running on Java 6 up until this day. With the end-of-life notification of the Flash runtime the server became obsolete quite quickly and the main reasons for hosting the server are a few functionalities which were not refactored into the new products after the migration and because of an ongoing SSO integration towards this server.

The next sub system of the Papirfly solution is the CHILI publisher installation which is self-hosted on the Windows servers within the AWS environment. The installation is quite IO demanding as the CHILI installation does not utilize any external database and works entirely within the filesystem of the given server, or in our case within a shared network disk space backed by AWS FSx. The server is being auto scaled depending on the length of the queue in the internal transcoding of this application. Additionally, the whole backend installation is being protected by two front-facing web servers which balance the incoming traffic between the auto scaled backend servers. The installation is supported by another third-party service installation called Pitstop, mainly used for PDF optimization. This service was introduced into the environment because of specific customer requirements on multiple of the PDF attributes from its size to a specialized color profiles required for certain print advertising vendors.

Finally, the Oracle REST Data Services servers are part of the legacy EC2 ecosystem. These servers just proxy requests from the CDN and allow the developers to call the Oracle stored procedures in a way a normal REST API would be called. This part of the solution is still one of the most used ones so there is proper load balancing and auto scaling set up on these servers as the load is ramping quite rapidly during the day. These servers are being load balanced based on the CPU usage provided by the AWS monitoring tools.

When we talk about the Legacy applications within the Papirfly ecosystem, we also must include the Oracle Database itself as it is the home for the heart of the whole solution, the Data Asset Management application. It is a central application which all the other applications within the Papirfly ecosystem are built upon as it stores all the original and further processed data and interfaces it to the end customer. The old solution was built solely using the Oracle stored procedures, but this solution is obviously put in sunset mode and a new solution was built using recent technologies. The solution had to be kept backwards compatible so because of this the new solution is still depending on the Oracle Database but only when it comes to data stored within the Oracle Database, all the logic from the stored procedures was or is being rewritten into mostly containerized workloads which consist of Java, Kotlin and PHP code.

7.2 Containerized applications

The entry point into the Papirfly backend ecosystem starts with the application called API Gateway which is a standard application which stands between the Internet and the internal platform. It takes care of actually exposing the given service to the internet and checks if the caller of the given endpoints is properly authenticated, it also keeps the session associated with the given called and it connects a JWT token with the session which is used for authentication further down the line when calling all the backends and APIs in the ecosystem. There is an intermediary service which is conveniently called JWT API which takes care of creating the JWT tokens with the correct information and privileges associated with the given user who is trying to log into the system. There is also a secondary API which is only internal facing and provides various legacy options of obtaining the JWT tokens. It is called Login API.

All the applications are following this API Gateway to Backend for Frontend pattern, but some APIs are additionally exposed via the API Gateway directly to the internet to be used with a generated JWT directly as multiple Papirfly customers are utilizing the system this way.

The main component of the whole ecosystem is the Data Asset Management, shorter for DAM, project which takes care of the management of all the static and dynamic graphical content ranging from simple images to video recordings spanning across multiple hours. The primary part of the application thus consists of a backend for frontend and the DAM API which supports all the relevant actions used inside the DAM application. Beneath the initial layer is the Transcoding application which takes care of transforming the uploaded material into acceptable forms specified either by the system itself or the customers. The system stores tens of thousands of specifications and these applications take care of flexible configuration of these specifications and providing the customer with the end results of these specifications. The application is made with Kotlin as it is a successor of the old Delphi Transcoders. The entire system is highly scalable as queue lengths might range from exactly zero to tens of thousands of assets being requested for specific transcoding versions. Another remarkably similar application supporting the asset information extraction after the initial upload or in on-demand fashion later is the Analysis bundle of microservices which either call a third-party analysis tool or an in-house made services depending on what the relative customer is setup with. This information is further processed and stored with the given assets and the information is then displayed in the DAM application to the end user. Most of the information about DAM Assets is stored in the SOLR search engine as the main purpose of the DAM is to search for the relevant information in a quick way. The SOLR search engine itself does not provide the necessary security features so the SOLR is being protected by an additional layer connected to the Papirfly authentication and additional rules limiting the result set of the SOLR search engine depending on the privileges of the given user. Next critical component of the DAM application is the Download API used for downloading the material, assets, and their transcoded variants. This part of the system is also highly scalable as it is prone to heavy load peaks during standard working hours. The service with the opposite functionality - called the Upload API - is the main point in the system where all the material is being uploaded into DAM. It is designed to be exceptionally durable and highly scalable as it is also a part of the system, which is prone to unpredictable loads, for example when a new customer is being onboarded.

The main DAM application also supports ingestion of material which is produced by various in-house or third-party tools. One of them can be CHILI publisher which is a third-party templating solution, there is also an in-house build video generator which operates with Adobe After Effects templates and provides a sensible UI where the user can generate various materials from the given templates. When the video generation is done the content can be managed and viewed in the DAM application. DAM is also being supported by various sync services which provide integration into external systems like Screen9, Templafy, Tactic or former Papirfly's templating solution.

DAM is also being supported by various applications which build upon the essential DAM workflows. One of the biggest applications is called GDPR and it takes care of all the consents and their expirations in addition to all the functionality required to be implemented by the General Data Protection Regulation adopted by the European Union in all the member states across Europe. Next application may include grouping functionality where the relative vendors or users can see template groups and take various actions in batches depending on the datasets provided or a My Creatives application which summarizes and tracks the relative template and creative generations across the entire system for the given user. The generations can be displayed inside the UI and the relative progress of the generation can be tracked using this application. One more supporting application which can be mentioned is the Color

Management application which serves as a data bank of colors for each customer and depending on their branding relative templates can be locked to use of the specified color palettes and so on.

Another bigger application built on top of the DAM now is the Collaboration tool which is used for approval loops before the assets can be distributed via the DAM interface into the other integrations. One of those integrations is the Brand Hub application which is a WYSIWYG editor for front pages of the given customer. The page has various editors, components, and integrations towards the whole Papirfly ecosystem, one of them being the DAM integration.

Shared Assets is an application which builds on top of the Brand Hub and DAM connection, and it is used to share assets from the Brand Hub. The application also tracks and controls in which ways the assets are being shared and which users the assets can be shared with.

The next small ecosystem is created by the BM2013 application which is a monolithic application made and released in 2013. It consists of multiple modules which are slowly being rewritten into microservices and some of them include functionality for ordering print advertising based on the various printing vendors which the application aggregate and a Marketing Shop functionality which provides ability for the Papirfly's customers to either sell marketing materials internally or to expose a store to the Internet and sell various branded items.

Campaign Manager is another part of the Papirfly ecosystem. The application is primarily used for scheduling events, either online or in-person, or for scheduling marketing campaigns. The Campaign Manager is a standalone and comprehensive tool which consists also of email WYSIWYG editor and has also support for sending out marketing campaigns via SMS and can be interconnected with other parts of the Papirfly ecosystem like DAM or Marketing Shop.

As one of the most important parts of the whole Papirfly ecosystem is to provide consistent branding across multiple platforms and applications, Papirfly must also have its tool to provide this consistency throughout the platform itself. For this purpose, there is an application called UI Builder which consists of all the styling throughout all the pages and applications in the Papirfly ecosystem.

There are also more backends and APIs supporting certain functionalities in the whole ecosystem and some of them are the Job API which is used for scheduling periodic jobs which can be done in specified intervals, the internal functionality is like the CRON functionality, but the application also exposes various metrics and tracks each job in a reliable manner. All the information can also be viewed in a handy dashboard accessible on the internal network. There is also a small but especially important part of the system, Toolbar, an application which is displayed on every page throughout the system and consists of various shortcuts and settings for the entire system. The last service mentioned is the Shorty API, used internally for shortening URLs. One use-case for the internal system is shortening links which are being sent out to the customers via SMS.

7.3 Latest serverless generation

One of the first applications from the latest generation of applications being built in a cloud native and serverless way is the Planner application being built on top of the AWS Lambda platform. The application itself should replace some of the legacy components within the Papirfly ecosystem and integrate it better with the platform. Next application from the serverless generation is the new reporting application which is also being built on the AWS platform entirely as it utilizes multiple managed services within AWS for Data Lakes, Data Warehousing and more. Finally, there is the new on-demand transcoding application which should take care of all the quick transcoders which are supposed to be done on-demand when the customer requests them. This form of transcoding also allows on-demand cropping and other image manipulation operations which are done on the fly during the request and then cached for extended periods of time.

7.4 Support Applications

There are also various third-party applications used in a containerized form. The main component used for most of the messaging within the platform is RabbitMQ which is used as a scalable cluster installation within the Kubernetes cluster in AWS. Next third-party application is called Aspose, and it is used for Word, Excel, and Presentation document transcoding and for generating thumbnails out of these types of files. For file scanning, one of the tools we utilize is the open-sourced ClamAV antivirus and as mentioned before the SOLR engine is crucial part of the DAM application and it is being hosted also in a highly scalable, durable, and available fashion inside the Kubernetes cluster in a containerized form.

8 INFRASTRUCTURE CONNECTING ALL THE APPLICATIONS RUNNING INSIDE THE AWS ENVIRONMENT

8.1 VPC

The different application environments are contained mostly inside the Virtual Private Cloud instance as this resource is the one responsible for virtually segregating the environment from the rest of the public cloud. It also hosts all the subnets and network configurations, and most of the computing and storage instances are connected to this inner network. The Papirfly organization chose the Stockholm region, referred to as eu-north-1 in the AWS system, which was chosen primarily for compliance reasons regarding data and where it is geologically stored. The Stockholm region offers three availability zones, so the infrastructure is spread across all these zones where there are three public subnets and three private subnets for the EC2 instances, and the network is also segregated further for the usage of the Kubernetes cluster. The public subnets have direct access to the internet through the Internet Gateway, the private subnets are routed through a NAT Gateway. The VPC also offers a lot of options to connect the services directly into the VPC so the calls within the network are not routed through the public Internet and are routed through the AWS backbone where it never the traffic is never routed to the public Internet. There are several in use and the primary include S3 service endpoint and Storage Gateway service endpoint because both mentioned services are the primary targets for majority of the data transfers in all the environments.

8.2 VPN

There is also a special service within the VPC service which solely takes care of the VPN connections to the internal network. The Papirfly ecosystem is still very much dependent on this type of connection into the internal network. To preserve the developer access to the relevant services there is one VPN Client endpoint which the developers can use to connect to the internal AWS network and access the necessary servers and databases without the need for exposing these servers or databases to the public Internet. It is worth mentioning this setup was marked as temporary in the beginning of the migration as there are tremendous costs associated with it as there are flat costs related to hourly charge on the VPN server running underneath and there is also a usage fee based on the number of connections in every given hour. Additionally, there must be a Route53 resolver associated with the VPN Client endpoint to propagate the correct DNS records to the clients via the OpenVPN protocol which incur additional costs.

8.3 Route53

Each of the application environments has two zones associated with it. One of them is for public use, and the second is reserved for internal use, marked as private, connected to the relative VPC in the given application environment. The DNS records in the private zone can then be utilized within the VPC to make the calls between the services more understandable.

8.4 Certificate Manager

Certificates are a central part of the Papirfly solution thanks to the multi-tenancy nature of the whole setup. Clients bring in their own domains and the solution needs to terminate the HTTPS connections using these certificates. The solution uses these certificates on the CloudFront edge locations and on a few given load balancers. The whole solution benefits from the automatic renewal of these certificates via a special DNS record setup which is used to verify the ownership of the domain and issuing the certificates for the given domain.

8.5 CloudFront

The CloudFront Edge Location is the first place any request towards the Papirfly solution hits on its path. It is used to heavily cache static content as the solution's main purpose is to provide or stream content in images or videos in various shapes and forms. Other requests may be routed to different backends depending on the application. The CloudFront distributions are also used to host the website domains for different customers and there is exactly one CloudFront distribution per customer as there is a hard limit rule imposed on the CloudFront service which prevents more than one certificate from being attached to a distribution and thus the Papirfly solution must create one distribution per its client domain which needs a separate certificate. One exception is the wildcard certificates where one CloudFront distribution can host up to 180 alternate domains which are valid for the given certificate.

8.6 Elastic Load Balancers

Elastic Load Balancers are the second layer which routes the request to the proper backends based on various rules specified within the load balancer configuration. The load balancers are used for the backend routings as there are numerous services within the AWS system where the applications can be hosted. Some of the target locations might include AWS Lambda Functions, EC2 Instances or the Kubernetes cluster pods. The Elastic Load Balancers are generally also involved in load balancing between the auto scaled services within AWS so separate load balancers are used for the specific parts of the Papirfly system which are prone to be the receiving end of traffic peaks such as the Delphi Transcoders which are scaled based on the length of the processing queue and the pool of the active transcoders can reach up to 50 instances at the highest traffic peaks. The load balancers are also used for health checking of the targets as otherwise the load balancers prefer to not route traffic to the unhealthy targets, and they send signals within the AWS system to notify the EC2 platform to replace the unhealthy targets within the auto scaling groups.

8.7 Storage Gateway

The Storage Gateway services are used as a replacement for the volume mounts which were used heavily in the former BrandMaster on-premises solution. This service emulates the volume mounts in the target systems and creates a caching layer above the S3 service which was chosen as the primary storage for most of the data during the migration into AWS. The service keeps this caching layer inside the memory, and it is backed by an EC2 instance which must have enough resources for loading all the bucket content into its memory.

8.8 S3

Simple storage service is being used for all the static files generated inside the Papirfly ecosystem, it is also used for most backups, logs and temporary files within the system which require fast access patterns. Rule of thumb used for the utilization of S3 is if the system does not specifically require it, it should use S3 instead of local or any other storage as S3 offers one of the best pricing opportunities and features when it comes to storage related services. The Papirfly environment is heavily utilizing the transitions and different tiers within the S3 service for various access patterns. There is also a specialized tier within S3 called Intelligent Tiering which automatically monitors the file access patterns and moves the files into the less expensive tiers if the file is not accessed within a specified number of days. There is an abstraction of the S3 interface in the form of mountable volumes via the Storage Gateway for the applications which are unable to be rewritten in a reasonable time.

8.9 EKS

The Elastic Kubernetes Service is used as the primary platform for hosting most of the applications in the Papirfly ecosystem. Most of the workloads within the system are containerized and are purely running inside the pods within the Kubernetes cluster. The only workloads not placed into the cluster are the legacy applications which were migrated in a lift and shift fashion because of the complexity and time it would take to rewrite them into a containerized form. The EKS cluster is interconnected with the AWS platform via various controllers running within the cluster. Some of them include the AWS Elastic Load Balancer Controller which controls the routing configurations for the load balancers based on the internal Kubernetes ingress configurations, or an External DNS controller which controls automatic DNS record creation from within the cluster when a service is created. The cluster is also connected to the EBS, EFS, FSx and Storage Gateway services for dynamically provisioning storage for different workloads.

8.10 RDS

Relational Database Service is also used quite heavily within the Papirfly ecosystem as there is a lot of relational data associated with all the assets and their metadata. There are three database clusters consisting of the central Oracle Database, MySQL and Postgres installations managed using the AWS Aurora distribution. The Oracle Database installation is made in a multi availability zone way, meaning the restarts are done by failing over to the standby instance and thus the restarts are almost seamless and without any prolonged downtimes. The Aurora clusters are structured in a read and write instance manner where there is one write instance and multiple read instances. The RDS service additionally takes care of the automatic backups on a regular basis and automatic maintenance windows where the databases are automatically updated if the instances are configured that way.

8.11 OpenSearch

The OpenSearch cluster serves as an interface for searching through logs aggregated from the entire system in all environments. The main purpose is dashboarding for traffic logs and application logs together with logs from the Simple Email Service which sends in send, bounce, and complaint logs. Those are critical when taking care of the emailing list in AWS as there are certain thresholds which when surpassed may result in termination of the given account.

8.12 SES

The Simple Email Service is used for sending all the emails from within the solution. There are diverse ways of setting the email inside the service itself. The setup in this environment is done mainly via configuration sets where all the applications are required to send in the configuration set which is configured for each customer and thus the usage can be tracked for the given customer and most importantly the amount of bounces and complaints are tied to a specific customer so if a customer has especially bad emailing list, an automated process can cut off this configuration set and stop more bounces and complaints coming in and making the solutions reputation worse. There is also additional setup needed if a customer requires to send emails from a custom domain which differs from the standard Papirfly domain used for generic or transactional emailing from within the solution. The setup is done via DKIM which is verified by adding special DNS records, generated by AWS, to the given domain.

8.13 SNS

Simple Notification Service is not heavily used for the application-to-application messaging, but it is quite extensively used for application to person notifications as the service supports many ways of notifications for developers and even customers. One of the features of the service is to deliver texts via SMS, which is a feature used in various applications throughout the Papirfly system to send out confirmation codes, event invitations and so on.

8.14 SQS

Simple Queue Service is utilized for standard messaging within the platform and prevents the applications from being overloaded with requests as the nature of a queueing system is to store the requests and the service itself pulls out the relative requests when it has enough capacity for it. The solution is utilizing the queues for the transcoding functionality and primarily for the on demand serverless transcoding as that part of the entire system is made in a cloud native and serverless manner, so it is utilizing as many managed services as it can. The queues are also being used for the new reporting platform as that is an event driven solution, so it naturally consumes a lot of data which needs to be queued in case of high utilization of the entire system and to preserve the durability of the whole platform.

8.15 ElastiCache

The ElastiCache service can be divided into two sub services. It is a service which provides an ability to run a managed Memcached cluster or a managed Redis cluster. These are two of the most recognized and used caches in the IT industry. The Papirfly platform is utilizing both. Memcached is used primarily with the PHP stack, Redis is used primarily for the Java, and Kotlin stack as it pairs nicely together with the ecosystem and its tooling for both mentioned variants of the caches. The caches are also important when it comes to sharing session information between multiple instances of any given service. The system APIs are primarily built to be stateless, so there is no need for the managed state in most of the solutions but there must be some front-facing applications which take care of these sessions. One of them can be the API Gateway for example. The service which takes care of the user login and all the routing to the inner system microservices. The ElastiCache solution is also used within applications themselves to cache heavy database queries on data which are not changed often and used for caching results of long running tasks.

9 INTERACTIONS OF THE DIFFERENT APPLICATIONS

The general interaction between the applications takes place in the form of an HTTP request or an asynchronous message sent through a message broker. The whole solution revolves around files so there are naturally a lot of interactions with either the local file storage or S3 over HTTP. The convenient place for the whole platform is the DAM application as that is the source of the truth for all the other applications. The first application the customer hits is the uploader application, integrated into the DAM. When the upload is done the file is stored in S3 where it is picked up by the transcoding processes and is processed further into relative formats defined for the given customer. After the transcoding is done the file can be further used in all the applications built above the DAM solution which can be diverse kinds of editors or automation tools for marketing purposes. All these actions are being asynchronously recorded or collected from the infrastructure logs and being sent into the reporting solution which provides the various metrics to the customers in form of customized reports and the data is also further analyzed for internal use.

10 ANALYSIS OF THE CANDIDATES FOR OPTIMIZATIONS

The migration's nature is always time pressing as the process incurs additional costs which double during the migration. Because of this, there are many compromises being made on the fly during the process and a lot of optimization work is being pushed further down the line. Computing is the most expensive component in most of the cloud users bills but the lift and shift fashion of migrating applications is the simplest and fastest as spinning up an EC2 instance and putting the application inside it is relatively easy, but this comes with the compromise of higher price tag until the application is rewritten or modified so it can be used with the cloud native components or at least containerized.

The Papirfly migration project had to also produce various compromises to speed up the migration process so the doubled expenses could be kept in a minimal time. There were a lot of lift and shift migrations done, especially for the legacy system components and there were also workarounds put together to work around the fact the solution is heavily dependent on local disks being mounted on the servers and containers. The S3 natively does not support this so the Storage Gateway service was chosen to provide this interface to the EC2s and the S3 buckets are being mounted as NFS shares onto the final servers and the applications not utilizing the S3 directly should be rewritten to interact with S3 directly as this service is an additional layer between the components and it should not be needed in the long-term perspective.

To combat uncertainty, the solution is also partially over-provisioned so the most logical optimization to be done is the right sizing of the relative instances and all the scalable components in the AWS eco-system. There are diverse ways of approaching the problem and it must be evaluated on the per component level as every component and application might react differently to changing the resources provided to it.

The overall state of the solution is also diverse, and the cloud solution could benefit from narrowing down the technological stack at certain points as the caching engines as the usage of more of them bring close to no benefit to the solution. There are also a lot of configurations which can be done in AWS to limit external traffic or usage of certain services to save costs.

10.1 Rewrite application storage to S3

The state of the already migrated solution is it is still dependent on the files being mounted as volumes on the underlying file system. This fact brings a lot of difficulties when migrating the files into AWS as the most cost efficient way to host files inside AWS is their Simple Storage Service, hosting files on a shared volumes like Elastic File Storage can get quite expensive in comparison the S3 solution with the fact that the S3 offers a lot of options for transitioning files between storage types base on the access patterns and so on. To overcome this technical limitation of the whole solution the Storage Gateway service was chosen to overcome it. It creates an abstraction above the S3 service which exposes the relative S3 buckets as NFS shares on the network and the given hosts can use these shares as mounts on the local file system. The solution can thus work normally without any significant changes to the application or any other added dependencies. This was used to lift and shift a significant part of the solution and a huge amount of time was saved using this method to avoid doubled costs when running a hybrid solution during the migration.

The natural suggestion for optimization in this case is to rewrite the applications themselves to utilize the AWS SDK to interact with the S3 directly and remove the intermediary Storage Gateway layer which abstracts the access to S3 in form of NFS shares which are mounted on the local file systems in every given host machine. This of course cannot be done all at once so this optimization is going to be an ongoing effort for an extended period. The good thing about this optimization is that the storage gateway itself is priced for data written to the S3 through the Storage Gateway NFS layer and it is priced as \$0.01 per GB. The terrible thing about this solution is the fact it is being backed by EC2 where it is not so scalable as the software itself is installed on the operating system inside the EC2 instance. There is a relative scaling which can be done to accommodate for the lowered traffic when a service is going to be rewritten to the direct access into S3 using the AWS SDK for the given language, but the costs of this solution are going to stay the same until the Storage Gateway is turned off. The optimization which can be done during the transition is therefore lowering the tier of the EC2 instance which is hosting the Storage Gateway service. The initial instance type and tier is m5.2xlarge which costs \$0.408 per hour in the Stockholm region. That equals \$300 per month if we standardize on the month having 30 days and a day being equal to 24 hours.

The optimization itself had to be done inside the DAM product as it is the primary source of the data related traffic inside the solution. The DAM consists of ten micro services which had to be rewritten to use the AWS SDK for Java to utilize the S3 module of the SDK. The application is now solely utilizing HTTP protocol to communicate with S3 and is completely independent of the local file system for uploading or storing files. All the logic related to files has been rewritten to utilize the S3 directly as thus the utilization of the Storage Gateway was lowered so there is not much data being written and the instance of the EC2 backing up the application could also be lowered to a lower instance which is still matching the hardware requirements. The instance itself was lowered to a m5.xlarge instance, which is the lowest possible instance the Storage Gateway can run on according to the official documentation. The data written through the Storage Gateway went from about approximately 1.5 TB a month to a mere 80 GB per month which is a significant cost saving for the solution as the traffic cost has become almost negligible as it usually does not surpass the $1 mark and the computing costs have been halved by lowering the instance.

The final suggestion for this case is getting rid of the Storage Gateway, but it cannot be done until every component in the system is rewritten to access S3 directly. This could not be done in the time span relative for this thesis, but it is a continuous effort being worked on and involves rewriting legacy components of the solution to utilized either S3 directly or utilize different abstractions of the S3 storage such as the s3fs which allows the Linux hosts to mount S3 buckets via FUSE, which stands for Filesystem in User space. This can be used for legacy systems which are difficult to rewrite and formally test. When all the components are independent of the Storage Gateway shares, it can be simply turned off which will bring the cost of this solution to $0 and thus eliminate all additional costs as all the costs are going to be centralized in the S3 service itself.

10.2 VPC endpoints

The essential component for the whole AWS environment is the Virtual Private Cloud which separates the cloud computing resources from the publicly exposed computing and puts it into a separated restricted area with dedicated networking possibilities as the customer can model the internal network to their needs. The standard architecture consists of public and private subnets where the public subnets are connected to the internet via the Internet Gateway service and the private ones connect to the internet through the NAT Gateway service. The NAT Gateway service is billed in both flat and elastic ways. The flat part of the billing is $0.046 per hour while running one of the NAT Gateway instances and the elastic part is also $0.046 per GB of processed data on the given NAT Gateway instance.

The problem in this case is most of the applications are being run inside the private subnets as none of the applications have the need to be directly exposed in the public subnet without any security measures whatsoever. Because of this fact all the application traffic must go through the NAT Gateway, including the traffic which targets the AWS services, like S3, as the only route to them is through the public internet at this point because there is no other route specified inside the private subnets. There is however a feature inside the Virtual Private Cloud service which is called VPC endpoints. This feature is used to expose endpoints inside the VPC itself and they can be referred to in the routings inside the private subnets. There are two options for the endpoints, one of them being the gateway endpoints which come free of any charge and the second one is the interface endpoints which are priced hourly. Each of them comes with their advantages and disadvantages but the gateway endpoints are a great option for the sake of optimization as they are free. The only gateway endpoints available are for the S3 and DynamoDB service but luckily these are the primary services which revolve around data and data transfers.

The optimization in this case is simply activating the VPC endpoints for the given services

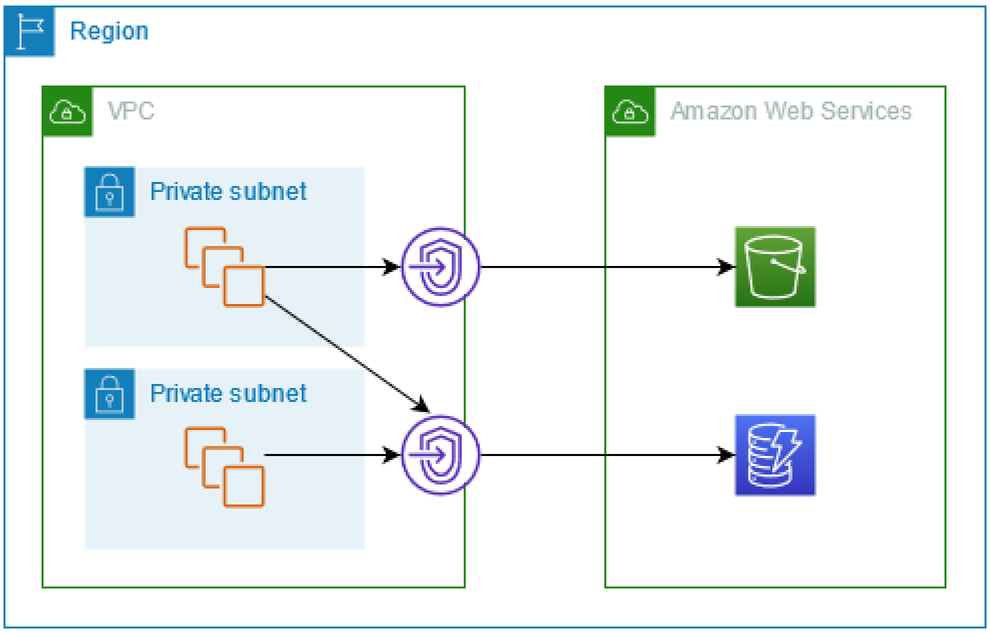

Figure 8 – VPC Gateway Endpoint [20]

and referring to them inside the private subnet route tables. From that point on the traffic is being routed directly to the S3 and DynamoDB service from within the private subnets and the traffic avoids the NAT Gateway service and thus the processing fees are also avoided all together. As the GB of processed data is priced at $0.046 and the standard monthly process volume is approximately 25 TB, the purchase price is approximately $1150 just for the data processing fees. By utilizing the endpoints, the data processing volume went down to approximately 4 TB which equals to a purchase price of $434. There are major savings on this because of the nature of the Papirfly platform, it is dependent on the S3 service because it is primarily data oriented as the main objective of the solution is to manipulate images and videos. Therefore, there can be a 62.2609% decrease in pricing for data processing fees just because the traffic is not going through the NAT Gateway service but straight to the S3 service by utilizing the VPC endpoint for the S3 service itself.

10.3 Using session manager instead of VPN

Developer access to the company's infrastructure and network is a major component in everyone's day-to-day work as there are several factors which can be factored into the need for this kind of access. First and foremost, it can be difficult to keep an updated dataset which can be used by the developers themselves on their local workstations so most of the times the remote access to a staging or a testing environment, where everything is setup in a similar way as the production development is setup, is critical for the continuity of the development cycle itself. There can also be several corporate level policies in place to prevent the system from leaking any confidential information outside of the given system and as such policies forbid any copies of the data being transferred to the local machines of the developers.

The Papirfly model was to use a VPN in order to connect to the internal services from the given offices all around the world, this setup is crucial not only for developers but also for other personnel as there are services which are not exposed to the public Internet, and it is required of them to connect to services like these via a secured channel like a VPN. Initially after the migration into AWS the VPC service was utilized to provide this kind of connection for everyone via the Client VPN sub service. The service works in an abstracted way, so the administrator of the system must create the service itself, associate it with the given VPC Subnets the end users need to connect to and then issue a certificate which must be imported into the Client VPN service. The administrator then needs to share this certificate with the end users so they can connect to the Client VPN endpoint via various VPN clients like OpenVPN.

The Client VPN service is priced at an hourly and hourly connection basis, meaning there is a general fee per active association to a VPC subnet and additional fee per active connections towards the Client VPN endpoint. These services are priced at $0.15 per endpoint association and $0.05 per connection hour to the Client VPN endpoint. The average connection hours usage could be estimated to be around five hundred hours a month so the total would be around $25 a month for the active connections from the local machines into the AWS network. The other part of the fee would be flat as it is priced hourly as $328.5 as a month has approximately 730 hours and there were three subnet associations needed to route to every subnet in use.

The proposed solution for this was to use the AWS service called AWS System Manager which consists of a subservice called AWS Session Manager, which is a service for making remote connections into the AWS infrastructure without the need for exposing any kind of ports to the public internet as it uses its internal service which is protected by the AWS authentication itself and then further forwards these connections via the HTTP protocol to the given infrastructure components which are able to accept these connections by using a special Session Manager Agent component which can be installed on all operating systems available in the EC2 platform.

The service itself offers direct connection to the EC2 instances so the service can be used directly when the only need is to secure a connection between a local machine and an EC2 instance. If, however the secured connection needs to be established between a local machine and any other AWS service there is a need for an intermediary component in form of an additional EC2 instance which serve as a Bastion or Jumpbox instance to relay the given connections and port forward the given traffic to other AWS services like RDS for example. In the Papirfly infrastructure there is such a need so there is a need for an EC2 instance which serves as a Jump box and forwards these kinds of connections. After several proof of concepts, a c5.large instance was determined to be the most appropriate for the number of connections the instance needs to handle on a daily basis. There is a need to have two of the instances as there is an additional need for managing access to different servers for different personnel. The c5.large instance is priced at $0.085 an hour so this translates to $61.2 a month for one of the instances and $122.4 for both instances needed to relay the traffic to the given AWS services like RDS.

This means the solution for the optimization brings down the costs for the operation by approximately 65.4% without taking the data transfer charges into consideration when it comes to both solutions.

10.4 Reducing number of Load Balancers

A core concept inside the Papirfly environment is routing as the whole solution is architected to run under one domain and there was not a need to think about prefixes in the URLs when the solution was hosted on prem. It is a completely opposite situation in AWS as the cloud resources which can route the incoming requests are limited. The solution is utilizing routing inside the CloudFront service for routes targeting different S3 buckets or different bucket prefixes but also different load balancers. The routing also differs on the caching and request policies. The second routing layer is the load balancers themselves. They are primarily used as application load balancers which route the requests based on different HTTP properties such as URI paths and host headers. The other option is to route the requests based on ports and for these kinds of scenarios the network load balancer is used.

The initial idea behind the load balancer setup was to let the application teams to handle the load balancers per application but the concept of per application load balancer became unrealistic quite early in the migration preparations as the Elastic Load Balancers are priced hourly for an instance of the load balancer running in the environment. Ever since then there is an agenda to keep the number of load balancers in the environment to the minimum as they incur additional costs.

The solution for this optimization problem is quite simple as you can group different applications under the same load balancer to avoid additional costs and thus cut the flat part of the Elastic Load Balancer pricing model. There is, however, a second part of the model where four key metrics are being evaluated on the given load balancer and these metrics are being translated into LCU units. The load balancers are measured on new connections, active connections, processed bytes, and rule evaluations. Each of these are evaluated into the LCU units in the following ratios - An LCU unit contains either twenty-five new connections per second, three thousand active connections per minute, 1 GB per hour for EC2 or IP targets or 0.4 GB per hour for Lambda function targets, or one thousand rule evaluations per second. Only the highest metric is considered when calculating the LCU units in the AWS billing.

The flat part of the billing inside the Elastic Load Balancer service is priced as \$0.02394 an hour for one instance of an application load balancer and the LCU part of the billing is priced as \$0.0076 per LCU hour. The network load balancers are priced lower on the variable LCU part and have their own LCU unit, prefixed with N - called NLCU. The flat part of the billing is the same as when it comes to the application load balancer. The NLCU unit, however, is priced at \$0.0056. For TCP, the NLCU unit includes 800 new TCP connections per second, 100000 active TCP connections (sampled per minute) or 1 GB of processed data per hour. For UDP, the unit includes four hundred new UDP flows per second, fifty thousand active UDP flows which are also sampled per minute or 1 GB of processed data per hour. For the TLS traffic, the unit includes fifty new TLS connections or flows per second, 3000 TLS active connections or flows which are sampled per minute or 1 GB of processed data per hour.

The original 23 application load balancers, which cost approximately $396 and $114 for the LCU units, were cut down to 6 application load balancers which cost approximately $106 for the instance hours and there was a slight increase in the LCU units as the load balancers get naturally more traffic, the LCU units are costs rose to approximately $190. The former solution's costs were about $510 and after the optimizations the costs are approximately $296, which is a 42.96% improvement when it comes to costs. The networking load balancers remain unchanged as the necessary routing is done on a port basis as the network load balancer does not have access to HTTP properties and can work only with the information the underlying protocol provides.

There are various considerations which were considered when it comes to specifics of the Elastic Load Balancer service. The Network Load Balancer scales very well with almost any amount of traffic so there are not any special considerations necessary, but when it comes to the Application Load Balancer there are various specifics to be considered. AWS is scaling the Application Load Balancer instances on-demand as the traffic flows through the service which means the load balancer is prone to increased response times when there are especially huge traffic spikes during peak hours, therefore the traffic needs to be thoroughly analyzed and spread across multiple load balancers accordingly so not all the most demanding system components are routed through the same load balancers. When these considerations were considered, the solution naturally evolved into six load balancers which serve traffic from the most important parts of the system and the other routes to the fewer demanding components are spread across these six load balancers.

10.5 EC2 Generations

When it comes to EC2 instances, there are various generations constantly innovated by AWS. One of the most groundbreaking news was the introduction of the Graviton processors made directly by AWS using the ARM architecture. The processor itself offers up to 40% increased performance and up to 20% decreased costs in the current generation so it naturally becomes a better alternative for better and cheaper computing.

There were several instances identified as potential candidates for the switch between these instance types. A lot of the cases inside the Papirfly environment were ruled out because of the increased complexity the migration between the generations and different architectures would cause. The ruled-out cases were legacy applications which are hard to migrate due to the lack of automation and substantial number of dependencies which would have to be manually checked and verified on the new processor architecture. On the other hand the chosen candidates were the exact opposite as the chosen candidates consist of managed services or newer application servers which are not directly dependent on the underlying operating system with a given processor architecture or applications which are written in environment independent and cross platform frameworks or languages such as Java or Kotlin applications which are also running in a containerized way inside a Kubernetes cluster which can also easily orchestrate these kind of changes.

The first obvious choice for the Papirfly environment was the OpenSearch cluster. It is a managed service, but it uses the EC2 platform underneath and it does not even try to abstract it away. The end user is presented with special OpenSearch EC2 instance types when creating the cluster. They are the direct equivalent of the standard EC2 instances with the OpenSearch components pre-installed on the underlying operating system which the end user does not have direct access to.

The OpenSearch cluster is being run with three dedicated primary nodes and six data nodes to keep the cluster highly available even during updates and potential reindexing. The primary nodes are using the c5.large.search instance type which costs $0.091 per hour and the data nodes are using the r5.xlarge.search instance type which costs $0.268 an hour. If we calculate the month to be 30 days and a day to be 24 hours the overall cost for the former cluster was approximately $1354.

The optimization in this case was quite straightforward as the OpenSearch is a managed service which has an excellent implementation of a blue/green deployment, which essentially means that the service itself spins up another environment, migrates all the data, switches the traffic to the newly created environment with the most recent changes of configuration and deletes the old environment. The changes for the cluster were to use the graviton instances for both node types in the OpenSearch cluster. The primary nodes were switched to the c6g.large.search instance type which costs $0.073, and the data nodes were switched to the r6g.xlarge.search instance type $0.214. All the nodes' final costs equal approximately $1082 which is an overall 20.0886% decrease in costs for the OpenSearch cluster alone.

Very similar improvements can be noticed in also in the Java/Kotlin application node group inside the Elastic Kubernetes Service cluster where the initial EC2 instances used were of the m6i.xlarge type and they were transitioned into the m6g.xlarge instance types and the approximate saving on these kinds of instances is approximately 19.6078% as the m6i.xlarge costs $0.204 an hour and the m6g.xlarge costs $0.164 an hour.

It is important to mention a lot of considerations must be considered as the OpenSearch service is directly compatible with this generation of the EC2 instances. Before migrating any service or application to the latest generation of the EC2 instances it should be considered that the migration itself can break the application or the service and the official documentation should be searched through before proceeding and updating the service. In the case of in-house made applications the ARM compatibility should be considered as various binaries and libraries will need to be recompiled or fetched for the right processor architecture.

10.6 Lambda ARM Generation

The same arm generation of processors is available on the AWS Lambda platform. The only thing to consider on the Lambda platform is the migration effort needed to migrate onto the ARM architecture. AWS Lambda offers a bunch of supported runtimes and all the out of the box runtimes should be quite easy to migrate if there are not any specific native dependencies which would require compilation for the ARM processor, which can be a complication for the Go and Rust custom runtimes.

In general, the lambda pricing is quite direct, and the costs are really decreased by approximately 19.9998% when tested on a small internal Lambda function. Most of the Lambda function spendings go towards the Lambda@Edge function executions which are Lambda functions run on the edge locations of the CloudFront service. This extended platform does not support the ARM architecture yet so there were no savings to be made directly inside the Papirfly platform itself.

The next generation of the Papirfly cloud native applications has still too small dataset of executions for any actual results but in the initial testing inside the Lambda spendings were genuinely decreased by the already mentioned 20% and the overall savings were in the range of a singular dollar.

10.7 Turning off stage for the nights and the weekends

One of the next major candidates for the optimization is the staging environment and its compute resources as the analysis of the whole environment shows that it is primarily being accessed during working hours as the main reason of this environment is for testing of new functionalities.

Most of the costs for this environment come from the static computing resources as RDS and EC2 where the EC2 is primarily occupied by the Elastic Kubernetes Service cluster where majority of the applications run and the legacy setups where most of the legacy applications run, on Windows instances. The RDS consists of MySQL, Postgres, and the Oracle server where the Oracle server takes up most of the costs. All these computing resources could potentially be turned off out of the office hours in the EU offices as the environment is not accessed at any other point in time otherwise.

The optimization solution lies in the AWS Solutions Library, which are already pre-made solutions running natively on the AWS platform, usually utilizing as many managed services as possible. The Instance Scheduler solution is a cloud native solution utilizing the automated scheduling of a Lambda function specifically searching for tags specified inside a DynamoDB table. The DynamoDB table also further specifies at which given hours and days of the week the services should be up and running, specifically the RDS and the EC2 instances. Based on this configuration the staging environment was configured to be up and running from 5 AM to 5 PM UTC with the databases starting and stopping 15 minutes before the initial startup so the unnecessary error logs are prevented. The solution is also configured to only start during working days, meaning the first time it is started is on Monday morning and it is turned off in the evening on Friday and it is kept turned off throughout the weekend. The Instance Scheduler solution is utilizing the EventBridge scheduling which can be easily turned off to keep the solution up and running which is a functionality frequently utilized during release periods or pen testing periods.

Before this optimization solution was implemented the approximate bill for the EC2 computing was $7000 dollars and the costs for the Relational Database Service was a bit higher and equaled $7500. The AWS Instance Scheduler solution with the mentioned configuration stably brought down the approximate monthly cost to $3000 for the EC2 computing resources and it brought down the costs to approximately $1000 for all the RDS resources which decreased the overall costs for the staging environment by approximately 72.4138%.

10.8 Usage of support

One of the most essential services any vendor in any space and market can offer is support. Services such as support are especially crucial in a time of need when production services and environments are impaired or when the business needs questions answered in a reliable and predictable time.

AWS distinguishes the Support into multiple tiers with different pricings. Some of the tiers can be activated by the end users and the Enterprise level must be communicated via the Support or the Customer Management team at AWS. There are the Basic, Developer, Business and Enterprise levels. The Basic, Developer and Business are the standard levels the root user of the account can turn between normally and the Enterprise is the one AWS must enable. The Developer tier costs either $29 a month or 3% of the monthly charges for the account. The Business tier costs either $100 a month or 10% of the monthly charges for the account up to $10.000, 7% up to $80.000, 5% up to $250.000 and 3% above $250.000. It is worth mentioning the service works on a subscription basis as upon activation you pay for the service together with the bill in the given month and the subscription is active for 30 days.

This is an essential thing when evaluating when do you really need the support and it is worth considering keeping the support service in an on-demand mode where the support is activated when it is really needed because otherwise the monthly bill can rapidly increase by the given percentage for each support tier.

The optimization here is tuning the support settings and setting it to the basic level and thus eliminating the costs for support each month. The approximate monthly bill before turning the tier down was equal to $1500. The tuning of the support tier eliminates this cost and as such it is a 100% decrease for this service as the service has not been utilized ever since.

10.9 Reduce usage of similar services

When migrating entire environments one of the major topics on the list should be reducing complexities of the given environments and one of the reductions might be for example be reduction on similar services inside the ecosystem such as cache systems as Redis and Memcached. The Papirfly solution was using both at the time of the migration and did not merge these two caching systems due to time constraints put on the migration project during the time. The final environment was running both caching systems in parallel in clustered forms. Upon the first inspection of the system, it was obvious that the system is not utilizing Memcached at all and only one of all the services within the Papirfly ecosystem is using the Memcached service.

The optimization for this was simple as the only thing which had to be done was the rewrite of the given microservice so it can be utilized together with the Redis service as that one is utilized in tens of microservices within the entire system. The final saving on this case was not enormous as the Memcached cluster used to be quite small but the final savings are equal to $40 a month and it is a 100% saving for the ElastiCache service as the Memcached cluster was eliminated from the infrastructure.

10.10 CloudFront Savings bundle

Another fantastic opportunity to utilize one of the AWS saving offers is the CloudFront service as this service is going to be heavily billed in all environments utilizing it solely because of the nature of the service which by design incurs traffic costs. After a certain period in AWS the accounts build up historical data which can serve as an indicator for estimating the sizes of the saving bundles.

The optimization on this case is also simple as the historical data suggested a commitment of approximately $2000 and the saving bundle saves approximately 30% of the traffic costs so in this case it is approximately $600 for the traffic charges inside the CloudFront service. The savings bundle also comes with a coverage of the Web Application Firewall charges which means another approximately $100 worth of charges falls under the CloudFront savings plan. The total amount saved is approximately $700 with the chosen savings plan.

11 COMPARISON OF THE STATE OF THE ENVIRONMENT BEFORE AND AFTER OPTIMIZATIONS

It is safe to say the migration of the whole infrastructure came with a lot of compromises due to the time constraints put on the project thanks to the doubled costs which were incurred by double hosting the environment on-premises and in the cloud during the migration process. The migration project was marked as successful and the system was fully able to utilize the offered cloud services in some shapes or forms, some of them emulated, some of them in the native form. The increasing bill became a problem because of the recent recession and therefore cutting down on business costs became essential for the longevity of most of the businesses.

The result of the optimization shows the migration was done in a quick manner with a lot of compromises the team had to deal with. These compromises could be optimized and could result in huge savings. Some of the savings included a simple configuration of some services like VPC Endpoints which are simple routings into the internal AWS network instead of routing to the AWS services through the public Internet. The other solutions were introduced in the form of re-architecting the former solution into a new one. Some of the optimizations were of a simple utilization nature as some of the services can be either used only on demand, such as the Support service, which is billed in a subscription model when activated, or the Saving Bundles which must be explicitly ordered.

Service optimization	**Former cost in $**	**Optimized costs in $**
S3 Rewrite	300	150
VPC Endpoints	1150	434
Session Manager Instead of VPN	328.5	122.4
Reducing number of Load Balancers	510	296
New EC2 generation for OpenSearch	1354	1082
Scaling down stage environment	14500	4000
Support usage	1500	0
Reducing number of services	40	0
CloudFront Saving bundle + WAF	2000	1300
Total	21682.5	7384.4
% Decrease		65.943%

1- Savings

As the table above suggests an approximate total of $14298.1 was saved by implementing the mentioned optimization and the bill for the chosen candidates decreased by approximately 66%. The major optimization which saves over half of the cost is the scaling of the staging environment which is not needed in the nights and the weekends and is only needed during the EU working hours as mostly only developers and testers are accessing the environments. The strategy implements scaling to zero which is possible thanks to the flexibility of the cloud providers and their billing models which are only charging for the active computing time and by turning off the computing power, the customers can eliminate the costs for computing completely.

CONCLUSION

The thesis shows the ever-evolving landscape of cloud computing offers businesses numerous opportunities to optimize their infrastructure and reduce costs, especially when utilizing industry-leading platforms like Amazon Web Services. By closely examining the cloud environments and implementing strategic measures, companies can achieve significant savings and maximize the benefits of cloud computing.

Rewriting applications for direct S3 communication are one such optimization that can lead to substantial cost reductions. This minimizes data transfer fees and reduces the need for additional compute resources, making it an efficient and cost-effective solution for data-heavy applications.

Updating the environment to newer EC2 generations is another essential optimization. Newer generations offer improved performance, better resource allocation, and can be more cost-effective overall. By staying up to date with the latest EC2 instances, businesses can leverage the most advanced technology while also benefiting from improved cost efficiency. An example can be Amazon's Graviton processor instances which offer up to 20% cost effectiveness.

Scaling down staging environments during off-peak hours, such as nights and weekends, can lead to significant cost savings as well. By adjusting the size and capacity of environments to match the demand, businesses can eliminate unnecessary costs and expenses and further optimize their resource utilization.

Utilizing on-demand resources is another crucial optimization tactic. On-demand resources allow companies to access additional compute and storage capacity only when needed, reducing overall costs, and making it easier to adapt to fluctuating demands. This flexibility is especially valuable in today's dynamic business environment. Some of the resources like support can really be activated only when needed as the cloud provider has a thought through business model around this service anyway and you are going to pay an appropriate compensation for their services even when activated on-demand.

In this business case, the combined effect of all the optimizations resulted in an impressive $14,298.1 reduction in costs and expenses. This clearly demonstrates the immense value in continuously evaluating and adjusting cloud deployments to maximize efficiency and minimize costs. As the cloud landscape continues to evolve, staying informed and being able to adapt is the key to leveraging these powerful platforms to their full potential.

Moreover, these cost-saving measures not only have a positive impact on a company's bottom line but also contribute to a more sustainable and eco-friendly approach to computing. By optimizing resource usage, businesses can reduce their carbon footprint and demonstrate a commitment to environmental stewardship.

In summary, the importance of regular cloud environment optimization cannot be overstated. By staying up to date with the latest technologies, adjusting infrastructure to match the demand, and harnessing the flexibility and elasticity of on-demand resources, businesses can unlock the full potential of cloud computing while simultaneously achieving significant cost savings. As we move forward in an increasingly cloud-centric world, proactive optimization will be a critical factor in determining the success and competitiveness of modern businesses.

BIBLIOGRAPHY

[1] PIPER, Ben a David CLINTON, 2019. AWS Certified Cloud Practitioner Study Guide CLF-C01 Exam. Indianapolis: Wiley. ISBN 978-1-119-49070-8.

[2] ARMSTRONG, Jeff, 2020. Migrating to AWS: Manager's Guide, Indianapolis: Wiley. ISBN 978-1-492-07424-3

[3] BURNS, Brendan a VILLALBA, Eddie a STREBEL, Dave a EVENSON, Lachlan, 2019. Kubernetes Best Practices. Indianapolis: Wiley. ISBN 978-1-492-05647-8

[4] PIPER, Ben a David CLINTON, 2019. AWS Certified Solutions Architect Study Guide. Second Edition. Indianapolis: Wiley. ISBN 978-1-119-50421-4.

[5] PERROTT, Sara a Brett MCLAUGHLIN, 2020. AWS Certified SysOps Administrator Study Guide. Second Edition. Indianapolis: Wiley. ISBN 978-1-119-56155-2.

[6] Overview of Amazon Web Services: AWS Whitepaper, 2021. Amazon Web Services [online]. Seattle, Washington, USA: Amazon Web Services [cit. 2021-04-28]. Dostupné z: https://d1.awsstatic.com/whitepapers/aws-overview.pdf

[7] RICHTER, Felix, 2023. Amazon, Microsoft & Google Dominate Cloud Market. Statista [online]. [cit. 2023-04-25]. Dostupné z: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/

[8] Secure and fast microVMs for serverless computing, 2023. Firecracker [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://firecracker-microvm.github.io/

[9] How Workers works, 2023. Cloudflare [online]. Cloudflare [cit. 2023-04-25]. Dostupné z: https://developers.cloudflare.com/workers/learning/how-workers-works/

[10] What is a cloud service provider?, 2023. TechTarget [online]. TechTarget [cit. 2023-04-25]. Dostupné z: https://www.techtarget.com/searchitchannel/definition/cloud-service-provider-cloud-provider

[11] What are cloud service providers?, 2023. Red Hat [online]. Red Hat [cit. 2023-04-25]. Dostupné z: https://www.redhat.com/en/topics/cloud-computing/what-are-cloud-providers

[12] Regions and Availability Zones, 2023. AWS [online]. AWS [cit. 2023-04-25]. Dostupné z: https://aws.amazon.com/about-aws/global-infrastructure/regions_az/

[13] List of Top 10 Microsoft AZURE Cloud Services in 2023, 2023. Rapyder [online]. rapyder [cit. 2023-04-25]. Dostupné z: https://www.rapyder.com/blogs/top-10-list-microsoft-azure-cloud-computing-services-2023/

[14] SAAFAN, Amr, 2023. Top 10 Azure Services in Demand. LinkedIn [online]. [cit. 2023-04-25]. Dostupné z: https://www.linkedin.com/pulse/top-10-azure-services-demand-amr-saafan/?trk=pulse-article_more-articles_related-content-card

[15] Products available by region, 2023. Azure [online]. [cit. 2023-04-25]. Dostupné z: https://azure.microsoft.com/en-us/explore/global-infrastructure/products-by-region/?rar=true&regions=

[16] Google Cloud overview, 2023. Google Cloud [online]. Google Cloud [cit. 2023-04-25]. Dostupné z: https://cloud.google.com/docs/overview

[17] The Top 5 Google Cloud Platform (GCP) Services for Businesses in 2021, 2023. LinkedIn [online]. LinkedIn [cit. 2023-04-25]. Dostupné z: https://www.linkedin.com/pulse/top-5-google-cloud-platform-gcp-services-businesses-/?trk=organization-update-content_share-article

[18] DENT, Kelly, 2023. AWS vs Azure vs Google: The battle for cloud supremacy. JeffersonFrank [online]. JeffersonFrank [cit. 2023-04-25]. Dostupné z: https://www.jeffersonfrank.com/insights/aws-vs-azure-vs-google-cloud-provider-comparison

[19] Benefits of using organizational units (OUs), 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/whitepapers/latest/organizing-your-aws-environment/benefits-of-using-ous.html

[20] What is Amazon VPC?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html

[21] AWS WAF, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/waf/latest/developerguide/waf-chapter.html

[22] AWS Lambda foundations, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/lambda/latest/dg/lambda-foundation.html

[23] What is Amazon DynamoDB?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

[24] What is AWS CloudFormation?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html

[25] What is an Application Load Balancer?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

[26] What is a Network Load Balancer?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

[27] What is Amazon EC2?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

[28] Amazon Elastic Block Store (Amazon EBS), 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html

[29] What is Amazon Elastic File System?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html

[30] What is FSx for Windows File Server?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/fsx/latest/WindowsGuide/what-is.html

[31] What is Amazon S3?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

[32] What is Amazon CloudFront?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html

[33] What is Amazon API Gateway?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/apigateway/latest/developerguide/welcome.html

[34] What is Amazon Route 53?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html

[35] What is Amazon EKS?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html

[36] What is Amazon Elastic Container Service?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html

[37] What is Amazon S3?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html

[38] What is Amazon Simple Queue Service?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z:

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html

[39] What Is Amazon EventBridge?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-what-is.html

[40] What is Amazon SES?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/ses/latest/dg/Welcome.html

[41] What Is AWS Certificate Manager?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/acm/latest/userguide/acm-overview.html

[42] What Is Amazon Kinesis Data Streams?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/streams/latest/dev/introduction.html

[43] What Is Amazon Kinesis Data Firehose?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html

[44] What is Amazon Relational Database Service (Amazon RDS)?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html

[45] What is Amazon OpenSearch Service?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/opensearch-service/latest/developerguide/what-is.html

[46] What is Amazon Athena?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/athena/latest/ug/what-is.html

[47] What is AWS Systems Manager?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/systems-manager/latest/userguide/what-is-systems-manager.html

[48] What is Amazon ElastiCache for Redis?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/WhatIs.html

[49] What is Amazon Transcribe?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/transcribe/latest/dg/what-is.html

[50] What Is AWS CloudTrail?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html

[51] What is Amazon CloudWatch?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html

[52] What is Amazon S3 File Gateway, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/filegateway/latest/files3/what-is-file-s3.html

[53] What is AWS Transfer Family?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/transfer/latest/userguide/what-is-aws-transfer-family.html

[54] What is IAM?, 2023. AWS Documentation [online]. Amazon Web Services [cit. 2023-04-25]. Dostupné z: https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

LIST OF ABBREVIATIONS

EC2	Elastic Cloud Compute
S3	Simple Storage Service
EU	European Union
AWS	Amazon Web Services
VPC	Virtual Private Cloud
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
TLS	Transport Layer Security
LCU	Load Balancer Capacity Unit
NLCU	Network Load Balancer Capacity Unit
NAT	Network Address Translation
URL	Uniform Resource Locator
SDK	Software Development Kit
NFS	Network File System
HTTP	Hypertext Transfer Protocol
DAM	Data Asset Management
PHP	Personal Home Page
API	Application Programable Interface
SQS	Simple Queue Service
SNS	Simple Notification Service
RDS	Relational Database Service
EKS	Elastic Kubernetes Service
SES	Simple Email Service
ACM	Amazon Certificate Manager
ELB	Elastic Load Balancer

WYSIWYG	What You See Is What You Get
UI	User Interface
JWT	JSON Web Token
CRUD	Create Read Update Delete
SOAP	Simple Object Access Protocol
RDP	Remote Desktop Protocol
WAF	Web Application Firewall
CDN	Content Distribution Network
IAM	Identity & Access Management
FTP	File Transfer Protocol
SFTP	Secure File Transfer Protocol
FTPS	File Transfer Protocol Secure
DNS	Domain Name System
VPN	Virtual Private Network
DDoS	Distributed Denial of Service
GB	Giga Byte

LIST OF TABLES