

Migrating company applications and services to the cloud

Michal Bernátek

Bachelor's thesis
2021



Tomas Bata University in Zlín
Faculty of Applied Informatics

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

Akademický rok: 2020/2021

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

Jméno a příjmení: **Michal Bernátek**
Osobní číslo: **A18036**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Softwarové inženýrství**
Forma studia: **Prezenční**
Téma práce: **Realizace přechodu aplikací a služeb jedné firmy na cloud**
Téma práce anglicky: **Migrating company applications and services to the cloud**

Zásady pro vypracování

1. Prostudujte a porovnejte vlastnosti cloud platform, jako např. Amazon Web Services, MS Azure, Google Cloud Platform.
2. Popište aplikace, které v rámci migrace budete nasazovat na cloud.
3. Proveďte a popište instalaci vybraných aplikací na danou cloud platformu.
4. Proveďte testování aplikace v běžném provozu.
5. Vyhodnoťte měsíční náklady na provoz aplikace v cloudu a porovnejte je se současným řešením.

Forma zpracování bakalářské 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 [online]. Indianapolis: Wiley [cit. 2020-10-19]. ISBN 978-1-119-49070-8.
2. SAVILL, John, 2020. Microsoft Azure® Infrastructure Services for Architects: Designing Cloud Solutions [online]. Indianapolis: Wiley [cit. 2020-10-19]. ISBN 978-1-11959657-8.
3. SULLIVAN, Dan, 2020. Official Google Professional Cloud Architect Study Guide [online]. Indianapolis: Wiley [cit. 2020-10-19]. ISBN 978-1-119-60244-6.
4. PIPER, Ben a David CLINTON, 2019. AWS Certified Solutions Architect Study Guide [online]. Second Edition. Indianapolis: Wiley [cit. 2020-10-19]. ISBN 978-1-119-50421-4.
5. PERROTT, Sara a Brett MCLAUGHLIN, 2020. AWS Certified SysOps Administrator Study Guide [online]. Second Edition. Indianapolis: Wiley [cit. 2020-10-19]. ISBN 978-1-119-56155-2.

Vedoucí bakalářské práce: **Ing. Tomáš Dulík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **15. ledna 2021**
Termín odevzdání bakalářské práce: **17. května 2021**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan



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

Ve Zlíně dne 15. ledna 2021

I hereby declare that:

- I understand that by submitting my Diploma thesis, I agree to the publication of my work according to Law No. 111/1998, Coll., On Universities and on changes and amendments to other acts (e.g. the Universities Act), as amended by subsequent legislation, without regard to the results of the defence of the thesis.
- I understand that my Diploma Thesis will be stored electronically in the university information system and be made available for on-site inspection, and that a copy of the Diploma/Thesis will be stored in the Reference Library of the Faculty of Applied Informatics, Tomas Bata University in Zlin, and that a copy shall be deposited with my Supervisor.
- I am aware of the fact that my Diploma Thesis is fully covered by Act No. 121/2000 Coll. On Copyright, and Rights Related to Copyright, as amended by some other laws (e.g. the Copyright Act), as amended by subsequent legislation; and especially, by §35, Para. 3.
- I understand that, according to §60, Para. 1 of the Copyright Act, TBU in Zlin has the right to conclude licensing agreements relating to the use of scholastic work within the full extent of §12, Para. 4, of the Copyright Act.
- I understand that, according to §60, Para. 2, and Para. 3, of the Copyright Act, I may use my work - Diploma Thesis, or grant a license for its use, only if permitted by the licensing agreement concluded between myself and Tomas Bata University in Zlin with a view to the fact that Tomas Bata University in Zlín must be compensated for any reasonable contribution to covering such expenses/costs as invested by them in the creation of the thesis (up until the full actual amount) shall also be a subject of this licensing agreement.
- I understand that, should the elaboration of the Diploma Thesis include the use of software provided by Tomas Bata University in Zlin or other such entities strictly for study and research purposes (i.e. only for non-commercial use), the results of my Diploma Thesis cannot be used for commercial purposes.
- I understand that, if the output of my Diploma Thesis is any software product(s), this/these shall equally be considered as part of the thesis, as well as any source codes, or files from which the project is composed. Not submitting any part of this/these component(s) may be a reason for the non-defence of my thesis.

I herewith declare that:

- I have worked on my thesis alone and duly cited any literature I have used. In the case of the publication of the results of my thesis, I shall be listed as co-author.
- That the submitted version of the thesis and its electronic version uploaded to IS/STAG are both identical.

In Zlin; dated: 15.5.2021

.....
Michal Bernátek v. r.
Student's Signature

ABSTRAKT

Téma bakalářské práce “Realizace přechodu aplikací a služeb jedné firmy na cloud” se zabývá business případem, kdy firma BrandMaster AS přesouvá veškerou svoji infrastrukturu ze svého vlastního hostovaného datacentra v Oslu v Norsku do cloudových služeb společnosti Amazon Web Services, zkráceně AWS. Cílem práce je analyzovat momentální trh spojený s poskytováním cloud computingu a popsat kroky, které bylo potřeba po dobu migrace provést pro udržení služeb i v průběhu migrace a kroky potřebné pro dokončení této migrace včetně otestování aplikací a porovnání nákladů na měsíční provoz celého dostupného řešení.

Klíčová slova:

Cloud, AWS, Amazon Web Services, GCP, Google Cloud Platform, Azure, Azure Cloud, BrandMaster, Migrace, Testování, Porovnání nákladů

ABSTRACT

This bachelor's thesis “Migrating company applications and services to the cloud” deals with a business case, where BrandMaster AS is migrating its whole infrastructure from self-hosted datacenter in Oslo in Norway to a cloud provided environment from Amazon Web Service, shortly AWS. The intention of the bachelor's thesis is to analyze state of the market connected to cloud computing and cloud services, describe the steps needed to keep the solution running throughout the migration process and also describe the steps needed to finish the migration as a whole, including application testing and comparison of the monthly expenses for maintaining the infrastructure.

Keywords:

Cloud, AWS, Amazon Web Services, GCP, Google Cloud Platform, Azure, Azure Cloud, BrandMaster, Migration, Testing, Comparison of expenses

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

I would also like to thank the CTO of BrandMaster AS, Morten Moe, for allowing me to use the Brandmaster's business case of moving the solution into the cloud as a topic for this bachelor's thesis and I would like to thank Jørgen Mølbach, the Senior developer in BrandMaster AS, for being a great colleague and helping me with formal questions, matters and materials used in this thesis.

CONTENTS

CONTENTS	6
INTRODUCTION	8
THEORY	9
1 CLOUD COMPUTING.....	10
1.1 AMAZON WEB SERVICES	13
1.1.1 NETWORKING.....	14
1.1.2 COMPUTING	15
1.1.3 DATABASES.....	15
1.1.4 STORAGE.....	16
1.1.5 OTHERS	16
1.1.6 PRICING.....	16
1.2 MICROSOFT AZURE	16
1.2.1 NETWORKING.....	17
1.2.2 COMPUTING	18
1.2.3 DATABASES.....	19
1.2.4 STORAGE.....	20
1.2.5 OTHERS	20
1.2.6 PRICING.....	20
1.3 GOOGLE CLOUD PLATFORM.....	21
1.3.1 NETWORKING.....	21
1.3.2 COMPUTING	22
1.3.3 DATABASES.....	23
1.3.4 STORAGE.....	23
1.3.5 OTHERS	24
1.3.6 PRICING.....	24
1.4 CLOUD COMPUTING OPTIONS COMPARISON	24
ANALYSIS	26
2 MIGRATION TO THE CLOUD	27
2.1 MIGRATION STRATEGIES	28
2.2 GOING INTO THE CLOUD	30
2.3 BRANDMASTER APPLICATION PORTFOLIO AND ARCHITECTURE OVERVIEW.....	32
2.4 LEGACY APPLICATIONS.....	33
2.5 MODERN APPLICATIONS.....	34
2.6 DATA.....	37
2.7 OTHER COMPONENTS IN THE INFRASTRUCTURE.....	38
2.8 APPLICATION CONNECTIONS	39
3 MIGRATION PLAN	41
3.1 FIRST PHASE	42
3.2 SECOND PHASE	46

3.3 THIRD PHASE.....	48
4 APPLICATION MIGRATION AND DEPLOYMENT.....	49
5 TESTING OF THE MIGRATED APPLICATIONS.....	51
6 MONTHLY EXPENSES OVERVIEW AND COMPARISON.....	53
CONCLUSION	55
BIBLIOGRAPHY	56
LIST OF ABBREVIATIONS	61
LIST OF FIGURES	63

INTRODUCTION

Cloud computing is a trend of the modern software development as more and more new developers want to focus solely on the application side of the development and they do not want to manage, provision and maintain any servers or any physical infrastructure. The serverless architecture is becoming one of the de-facto standards when it comes to developing applications and software. A business case was introduced shortly after hiring new CTO Morten Moe to BrandMaster AS. The business case described how the BrandMaster solution could save a major amount of money, become more secure and modern by moving its operations and workloads into the cloud.

The thesis explores the state of the market of cloud computing providers available. The first chapter is evaluating and analyzing their offers, pros and cons. The second chapter describes the business case of BrandMaster AS and all the steps the development team assigned to the migration process to keep the whole infrastructure, solution, and services up and running during the entire process of the migration. It explains the strategies and decisions taken before, during and after the migration which got the solution through the whole migration process. Furthermore, the thesis describes the former architecture and applications of BrandMaster solution and ties them to the described migration strategies.

The important part of the migration is testing of the migrated application and comparing the results with expectations. The scope of the thesis also compares and describes some reasonings and expectations regarding the hybrid solution which had to be made during the migration process due to the need to keep all services running as smoothly as possible for the end-users and customers of BrandMaster.

The last and final most important point is a comparison of expenses needed to maintain the critical infrastructure up and running all times in addition with workloads tied to a specific amount of time needed to finish the given workloads and tasks. This is the area where cloud computing certainly accelerates as almost all the cloud computing providers are providing pay as you go model where the customer is paying only for the resources which are actually used during the workloads. This ability to use only resources needed to get the workloads to finish together with additional optimizations can provide significant decrease in operation costs and expenses.

I. THEORY

1 CLOUD COMPUTING

Cloud computing is a relatively new paradigm which refers to cloud providers selling, providing, and managing services, infrastructure, platforms, or software in form of services provided to whoever is willing or wants to pay for them. [6]

The majority of the cloud computing market shares are divided between Amazon Web Services, Microsoft Azure and Google Cloud Platform. AWS with leading advantage of 32% worth of market share, followed by Azure with 20% and Google Cloud with 9%. The whole cloud computing market revenue is estimated to \$129 billion. There are of course other cloud providers with less market share than that. Some of them might be Alibaba Cloud, Oracle Cloud, IBM Cloud or Tencent Cloud. [7]

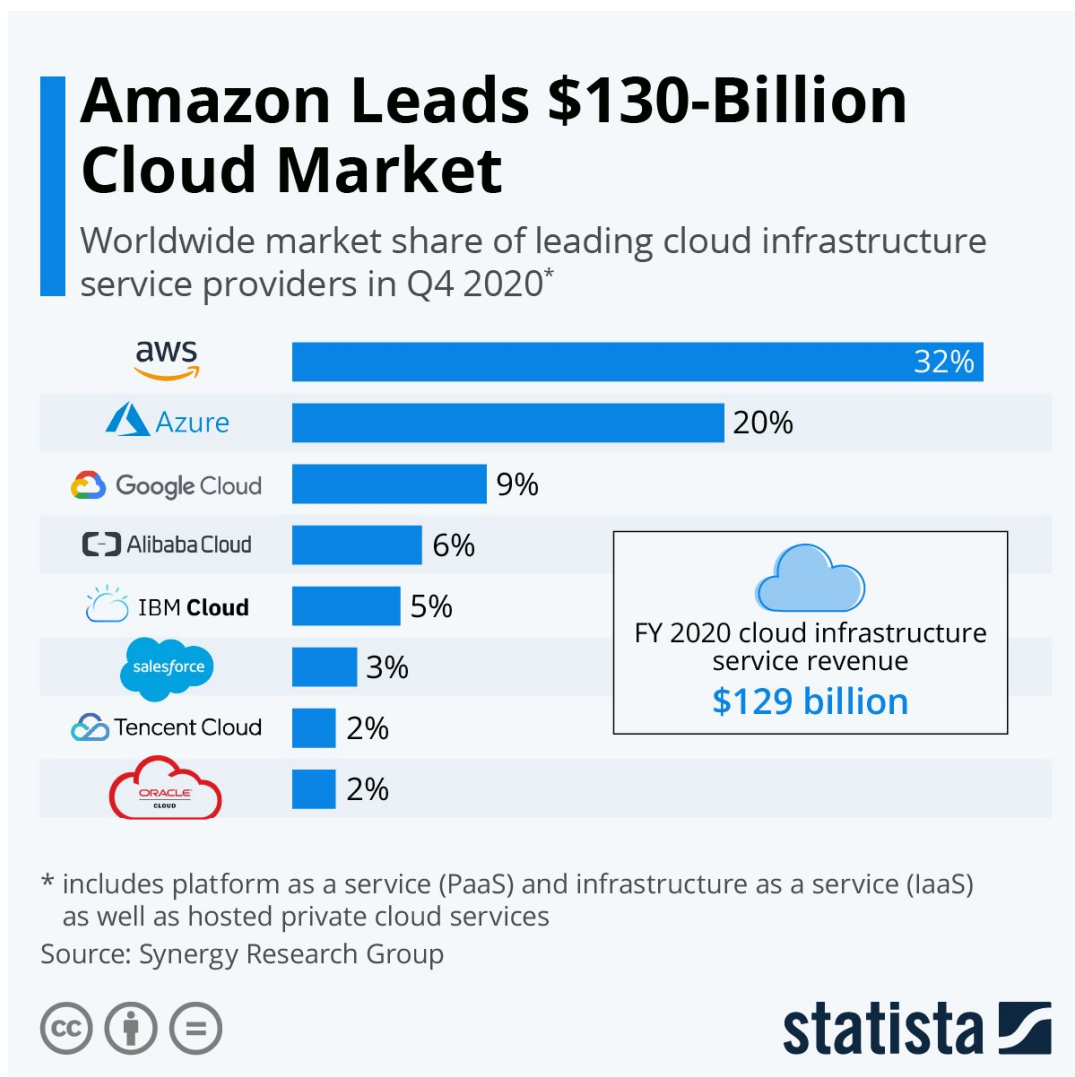


Figure 1 Market share of the leading cloud infrastructure service providers [7]

The main advantage of cloud computing usually is that the company or person who chooses to use these kind of cloud services is paying only for what they really use, meaning they only pay for the computing power which they really use. This billing model is often referred to as “on-demand” or “pay as you go”. This can be beneficial for any range of companies or corporations because it is probably the most efficient billing model for end user who pays for the services. Of course, the billing model depends on the given cloud provider and on the given service which is provided. Some services are billed by the hour, some by the minute and some can be billed in by the milliseconds. There are also provisioning billing models where the party ordering the services sign up for a certain price for using specified amount of any given computational power in the purchased service. Some cloud providers often offer convenient packages which come with a discount for signing up for them. [6]

There are also several types of cloud services or modules the customer can use based on level of abstraction the cloud provider provides or which the customer chooses to use. In some cases, there are multiple types of cloud services offered and the customer can choose which one to use. In general terms the levels of abstraction can be explained as what parts of the infrastructure or service is managed by the customer and what parts are managed by the cloud provider. Some of the reasons for customers may choose to use cloud infrastructure and cloud computing can be the fact they either do not want to manage their own physical infrastructure or do not have any qualified staff which would take care of the infrastructure. This is addressed by various levels of abstractions which provide the customer with the modules they need according to the qualification of the staff or according to the needs of the given customer. [8]

Let us say the traditional IT presents us with nine levels which we must take care of, dedicate qualified personnel to, and ultimately manage ourselves. The levels are: Networking, Storage, Servers, Virtualization, Operating Systems, Middleware, Runtimes, Data, Applications. Different abstractions shield the customer from different levels of this classical IT tower. There is Infrastructure as Service, shortly IaaS, which moves the need to manage the lowest levels of infrastructure to the cloud provider, meaning Networking, Storage, Servers, and Virtualization. Platform as a Service, shortly PaaS, which leaves you only with management of the data and your applications. Everything else is taken care of by the cloud provider of your choice. The next one is Serverless, which is a popular buzz word as of late. The architecture was created so the developers can concentrate only on the actual development of the applications. Everything else is done and taken care of by your cloud

provider. The last tier in the options is Software as a Service, shortly SaaS, which is basically running in the cloud, either made by you as the customer or made by the cloud service provider. Its usual interface is a website, where you get automatic upgrades and as it is running in the cloud you cannot really lose any data, provided that the cloud storage module does not fail. [9]

Cloud Computing Services: Who Manages What?

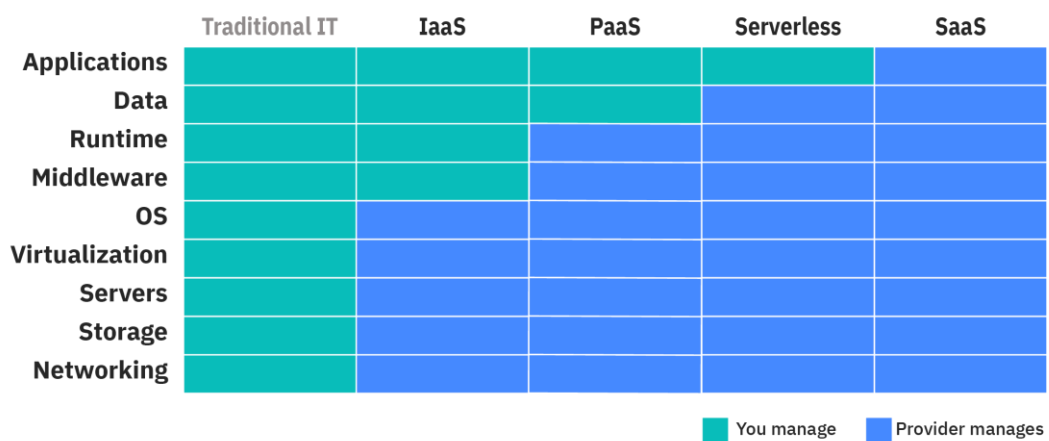


Figure 2 Illustration of management needs in different levels of infrastructure abstraction [9]

The next important division of the cloud space is the tenancy of the cloud computing environment. The customers can choose to use a public cloud, where the computing resources are shared in a multitenant way, meaning some workload can be computed on the same physical infrastructure by multiple customers at one given point in time. The cloud provider usually takes the responsibility for the datacenters, its infrastructure and performance with high-bandwidth connectivity and highly accessible data for the applications. On the other hand, private cloud is an environment assigned to one and only customer where the customer takes advantage of the benefits of scalability, elasticity, and ease of delivery, while they keep their solution on their premises or on a rented physical infrastructure managed by the cloud provider. The private cloud is often chosen because in a lot of cases it may be the only way to meet regulatory compliance requirements, or the customers need to meet the requirements given by the contracts signed with their clients. There also may be confidential documents or personal data used in computations in the cloud. These two types of clouds are totally separated as the naming suggests so there is a

need for a solution to connect these two. The solution is called a hybrid cloud. It connects the private and the public cloud. The customer decides which workload or services to run in a public section and which ones in the private section. This solution can lead to very flexible, effective, and cost-efficient cloud environment which takes advantage of all the positives of cloud computing from one cloud provider. [9]

The question here might be if it is not better to utilize multiple cloud providers. The answer is – yes, it is. Multiple companies and corporations answered that they have been using hybrid multi-cloud environments. Multi-cloud is an environment where the customer uses services managed by multiple cloud providers. The obvious benefit can be avoiding vendor lock-in, access to more services which may provide more innovation and less restrains on developers and software engineers, providing them with more freedom and letting them build more innovative and cost-efficient solutions. The downside of this environment is that it can become increasingly more difficult to manage these kinds of multi-clouds.[9]

1.1 Amazon Web Services

Amazon Web Services, shortly AWS, is a cloud computing platform provided by Amazon. Its offer consists of different Infrastructure as a Service, Platform as a Service, Serverless and/or Software as a Service modules which can be used to build your own cloud native software. AWS was launched in 2006 and it was created mainly from the internal infrastructure which was used to run and handle all the computing for Amazon.com. As one of the first companies it introduced the “pay as you go” billing model which provides better scaling for each customers’ needs.[10]



Figure 3 Example of the AWS interface. [12]

1.1.1 Networking

One of the core components of Amazon networking structure is a module called Amazon Virtual Private Cloud, shortly Amazon VPC. It lets you configure your own isolated network structure, including selection of your IP address ranges, public and private facing subnets and configuration of routing tables and network gateways. [10]

Amazon Route 53 is very scalable and highly available DNS service. One of its biggest advantages is that it was designed to be extremely cost-effective service to route users to applications by translating the readable names and domains into IP addresses or point directly to AWS services such as EC2 instances, Elastic Load Balancers or Amazon S3 buckets. There are also highly advanced functionalities like DNS health checks, Geo DNS, weighted routing, or DNS failover. With all these functionalities combined customers can create highly fault-tolerant systems with low latency. [10]

Elastic Load Balancing can distribute incoming traffic across multiple EC2 instances, containers or IP addresses inside your Amazon infrastructure. It offers high availability as the load balancers can be available across multiple availability zones and they can also scale automatically according to the incoming traffic. There are three types of load balancers in AWS. Application Load Balancers are HTTP and HTTPS load balancers which are targeted at traffic delivery for applications, microservices and containers. Network Load Balancers are suited for load balancing of TCP traffic where the primary requirements are extreme performance and ultra-low latencies. Classic Load Balancers were created for basic load balancing across multiple Amazon EC2 instances.[10]

Amazon CloudFront is a huge and fast content delivery network service which can deliver content including images, videos, application data and API responses on a global scale, with high transfer speeds and low latency. Thanks to its Edge locations spread all over the world it can leverage its caching mechanism to deliver non frequently changed content even faster. The service also integrates very seamlessly with other networking services and services which aim to protect the infrastructure from external attacks. [10]

API Gateway is a fully managed service which aims to make it easier for developers to work with APIs. It was created to primarily create, publish, maintain, monitor and secure APIs and provide scaling for all customers' needs. [10]

1.1.2 Computing

Amazon Elastic Compute Cloud (Amazon EC2) is the service which stands behind most of the computing inside AWS eco-system. It provides virtual server instances which can run workloads. EC2 provides diverse range of instances based on different parameters associated with the given EC2 instance including RAM, vCPU, network speed, graphical cores and storage options.[3] There are also multiple instance types which are divided according to the billing model the customer chooses. On-Demand instances are paid by the hour with no long-term commitment. These kinds of instances are especially useful for example for unusual and/or periodic traffic spikes using automatic scaling. Reserved instances come with discounts up to 75% but you are committing to a fixed duration you are renting the instances for as a customer. Spot instances comes with up 90% discount and they, as AWS describe, use spare capacity in the AWS EC2 service. You can specify your maximum price per hour you, as a customer, are willing to pay and your workloads can spin up these instances when the spot price goes below the specified value. [10]

There are two main products which enable you running containers in AWS. One of them is Amazon Elastic Container Service which is a container orchestration service build by AWS. It allows you to run docker containers and eliminates your need for managing the host environment of Kubernetes or Docker Swarm clusters. Next one is Amazon Elastic Kubernetes Service which was created for customers which want to manage their own Kubernetes cluster inside AWS. [10]

AWS Lambda is a serverless function service which lets you run code without provisioning or managing any servers.[10] There are no costs when the code is not running and you pay only for the execution time of your code, newly measured in milliseconds since 2021.[11] Lambda functions can also be invoked from a most of the services as a consequence of an event in the AWS environment.

1.1.3 Databases

Primary database service in AWS is called Amazon Relational Database Service, shortly RDS, and as the name suggests it allow you to run different relational databases with different types of underlying server instances based on your preferences and needs. The supported database engines include Amazon Aurora, PostgreSQL MySQL, MariaDB, Oracle Database and SQL Server. The RDS service also provides functionality for hardware provisioning, automatic database setup, patching and backups. [10]

1.1.4 Storage

AWS has multiple services for different kinds of storages. The main one is called Amazon S3, it is an object storage which can be used for wide range of use cases like hosting data for websites, mobile applications, IoT devices and big data analyzation. It can also be used for data backup and its restoration together with Glacier service which extends the S3 functionality. For EC2 instance storage the Amazon Elastic Block Store is where you can provision volumes which are automatically replicated across availability zones. For Linux-based external storage which can be shared between multiple EC2 instances, there is a service called Amazon Elastic File System. It is a service created for massively parallel workloads and it can automatically scale to petabytes and back as you add or remove files without any downtimes for connected applications. [10]

1.1.5 Others

AWS offer consists of very diverse range of services for IoT, Machine Learning, AI, Robotics, Game Development, Blockchain, Quantum Technology or even Satellite modules which can be used in different levels of software development and can be used create wide variety of systems. AWS is also still developing new managed services to expand their already broad range of services. There are also ready to use applications like Amazon Chime for Meetings, Video Calls and Chat or Amazon WorkMail for managed business Email and Calendars. [10]

1.1.6 Pricing

One of the biggest disadvantages of AWS which is being talked about is the pricing and its large variety. Almost all of the services come with its own pricing rules and billing periods, so it is often matched as confusing and even though AWS is offering its calculator, so the customers can create a relatively accurate estimate, the overwhelming variables in the calculator make the calculating process inscrutable. [13]

1.2 Microsoft Azure

Azure is a cloud computing platform provided by Microsoft. It features more than 200 services which can be used to help you build new solutions. Azure also provides different levels of various services including Infrastructure as a Service, Platform as a Service, Serverless and Software as a Service. It was initially released in 2010 and it has been

expanding its product portfolio with new features in all the important and relevant fields. This platform has the most support for Windows based products like Office software, Active Directory and different Windows servers but you can also run Linux based workloads. [14]

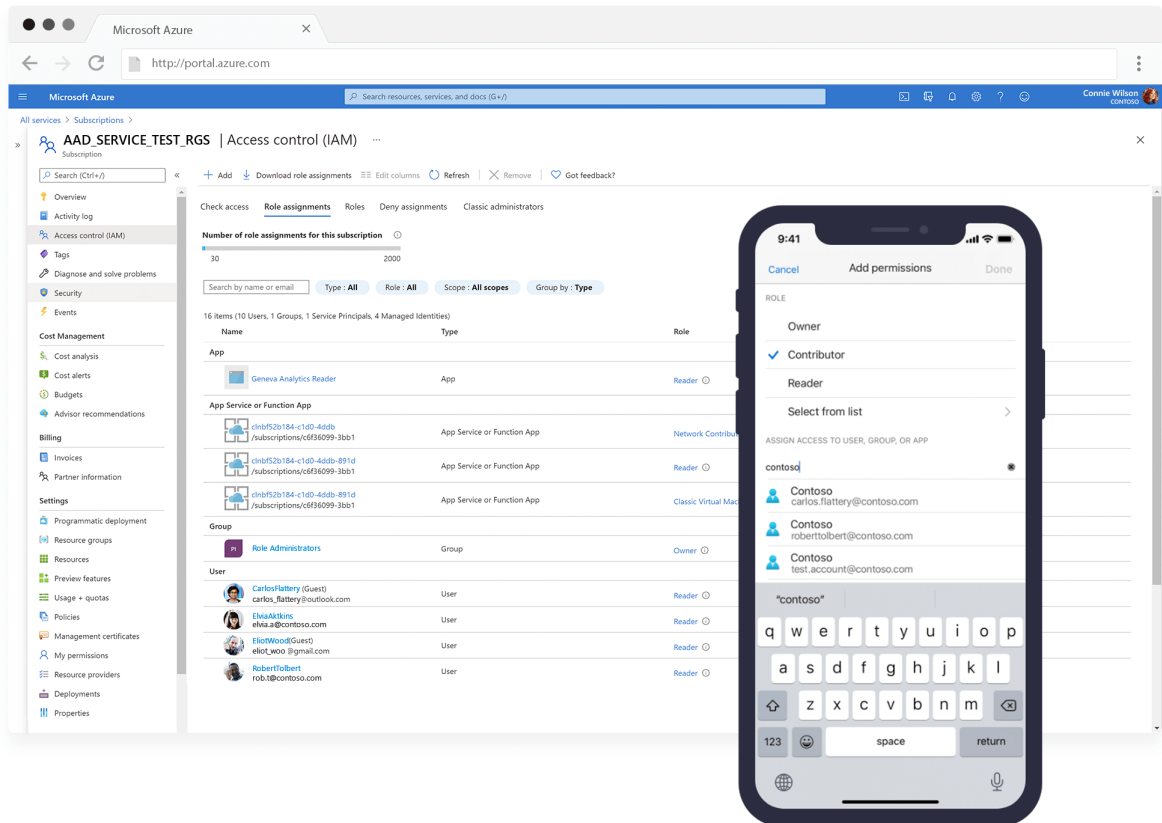


Figure 4 Example of the Microsoft Azure interface[15]

1.2.1 Networking

Virtual network, shortly VNet, is one of the core services in Azure. It allows you to build private networks where azure resources or even networks can communicate between each other. Azure resources which can be deployed into the given virtual network include VMs, Azure Service Environments or Azure Kubernetes Service.[2] Networks can be spread in different azure regions and can use virtual network peering to connect. An essential part of the virtual network is the ability to connect to the internet. All outbound connections are allowed, by default, and inbound connections can be established after assigning a public IP address or a public Load Balancer. Next important feature can be communication with on-premises networks. This feature is enabled either by VPN Gateway or ExpressRoute, these

services will allow you to create private connections to the customers' internal networks. [16]

Azure DNS is Azure's variant of a DNS hosting service. If you decide to host your domain inside the Azure environment, then you can use and manage it using the Azure integrated services, credentials IPs, tools overall and you can include the billing for your domains in your invoice, same as all the other Azure services. [16]

Azure Traffic Manager is a service which provides traffic load balancing on a DNS level. It allows you to send traffic and distribute it among different Azure regions or services. All of that with high availability and quick response times. It enables you to route traffic in different ways including priorities, weighted routing, performance, geographic location and it also allows you to route based on multi-value and different subnets. [16]

Azure Load Balancer is a service for high-performance and low-latency load balancing for all UDP and TCP protocols. It manages inbound and outbound connections, you can configure public and internal load balancing endpoints and you can set different rules to route connections to specified destinations, which can be monitored with HTTP health probing. [16]

Azure Content Delivery Network, shortly CDN, is a global solution for content delivery all over the world by caching the transferred resources on the physical nodes spread across the planet. The cache is stored on edge servers which are in point-of-presence locations. These locations represent the nearest location with the lowest latency for the end user who is requesting the given content. [16]

Azure Front Door is a service which allow you to manage routing for you web traffic and specify routes pointing to different services in different regions. It also features failover to enable you to have the best availability. [16]

1.2.2 Computing

Azure Compute is the heart of the computing in Azure. It allows you to run computing resources in the cloud. You can choose from multiple different disks, processors, RAM sizes, network speeds and operating systems. Azure also supports wide range of computing solutions made for application development, application testing and running applications. It also supports multiple levels of the infrastructure services including Azure Virtual Machines, Azure Container Instances, Azure Kubernetes Service and Azure Functions. [17]

Azure Virtual Machines are software emulations of physical servers. They have their own virtual resources like a virtual processor, RAM, storage and network resources and each virtual machine has its own operating system of your choice. There is also a wide range of variants of the virtual machines with different resource parameters, optimized for various tasks and depending on how much resources does the task need. There are also saving plans in place so the customers can save additional money when the plan suits their needs. There are reserved instances which can save up to 72% when compared to pay as you go pricing and if you combine it with Azure Hybrid Benefit, which takes advantage of smart re-use of on-premises licenses when running Windows servers, you can save up to 80% when compared to pay as you go pricing. There is also spot pricing available, and customers can buy unused computing power at deep discounts. [17]

There are services for running containers inside the Azure, called Azure Container Instances and Azure Kubernetes service. Both services are an easy way to run containers, manage them and scale them dynamically and fast. Azure Container Instances is a service which provides a way to run containers without the need to manage any servers or clusters. It gives the customers an ability to fully focus on designing and making the actual applications. Azure Kubernetes Service on the other hand gives you a fully managed Kubernetes service. It lets you run containerized applications on Kubernetes cluster you as a customer manage yourself. You do not manage the physical infrastructure, but you essentially tell the cluster how to operate, you scale the cluster by provisioning additional virtual machine instances and so on. [17]

Azure function is the main serverless service in the Azure cloud. It is primarily used when you want to only take care of the code of the application and do not manage the infrastructure at all. It is often used in conjunction with events such as REST calls, timers or messages from the other Azure services or when the task is required to complete fast. [17]The supported language runtimes include C#, JavaScript, F#, Java, PowerShell, Python and TypeScript.[18] The service is billed per second and also offers 1 million of executions per month.[19]

1.2.3 Databases

Database repertoire of the Azure cloud includes one non-relational database called Azure Cosmos DB and multiple SQL ones like Azure SQL database, Azure Database for MySQL, MariaDB and PostgreSQL. All of these services are fully managed so you as a customer do

not manage any infrastructure, but there are multiple offerings where you can manage the database on the operating system level. The benefit of managed service is that the database is always up to date, the service manages maintenance, backups and scalability of the database clusters. [20]

1.2.4 Storage

Azure Blob Storage is a service optimized for storing huge amounts of unstructured data such as text files or binary files. Some of the use cases for this service might be video or sound streaming, image or document storage for HTTP calls, storing data for backups and storing data for local analysis by other Azure services. Customers can access the stored data through HTTP or HTTPS from around the globe using URL addresses generated through the Azure interface, by using the Azure REST API interface, Azure in PowerShell, Azure CLI or client-side libraries for Azure Storage available for .NET, Java, Node.js, Python, PHP and Ruby. [21]

Azure Files is providing highly available directories shared over the network. The directories are primarily available over SMB protocol which means multiple servers can share the same files with same access rights. The files are also available through a REST interface or client-side libraries. [21]

Azure Disks is an abstraction above the blob storage which emulates virtual disks for virtual machines in Azure. It is usually used when you really want a simple local disk storage attached to the virtual machines and you do not want to share the data with any other service other than the local machine. [21]

1.2.5 Others

Azure is a leader when it comes to Windows and Active Directory, they own the services after all. Azure also offers a lot of Machine Learning and AI capabilities, IoT services and solutions. There is also a big variety of analytics services available in the Azure cloud environment. [17]

1.2.6 Pricing

Azure has a more clearly laid out billing structure when compared to AWS. Each service has more similar pricing rules which make it more readable for the end users. It offers a well-arranged calculator which can summarize the costs of all included services. When it comes

to pricing itself, it offers mostly per second billing, but not all services offer this kind of pricing yet. [13]

1.3 Google Cloud Platform

As the name suggests, Google Cloud Platform is a cloud computing platform run by Google. The initial version of the platform was released in April 2008 as a preview version of the App Engine[22] and in November 2011 the app was released as a fully-fledged version of the application. [23] Since then, Google has listed more than 100 products. [25]

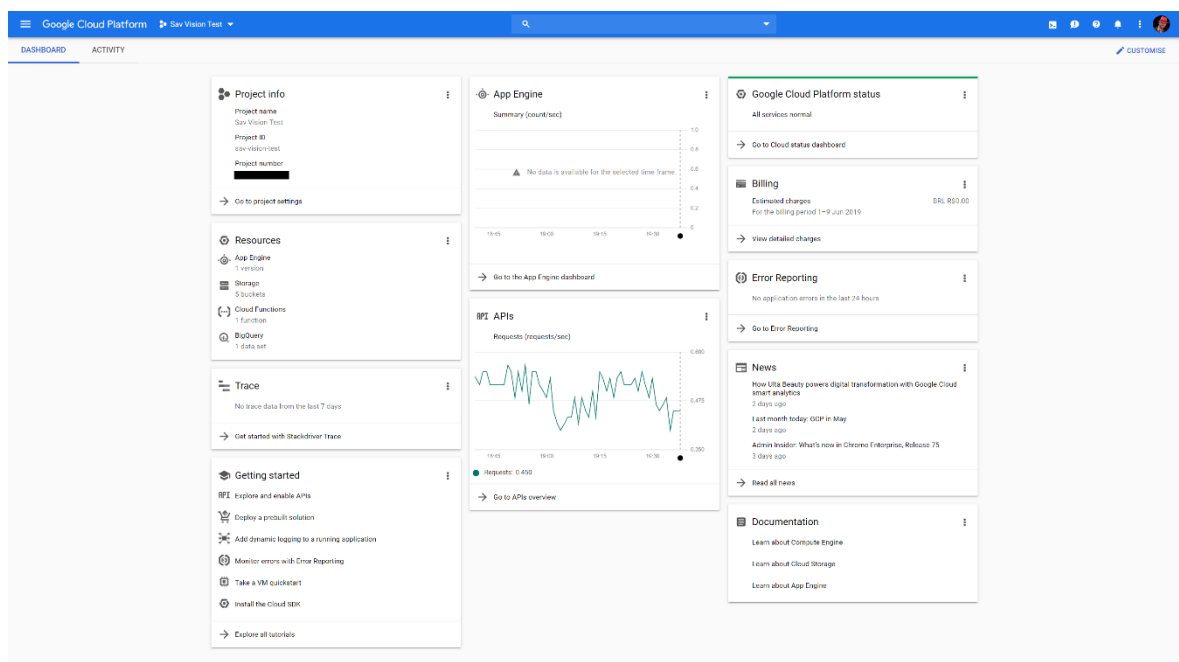


Figure 5 Example of the Google Cloud Platform dashboard [24]

1.3.1 Networking

Virtual Private Cloud is a component which provides different range of networking services which can be used in conjunction with VM instances on GCP. The services allow you to create your own network topologies and your own IP allocations or segmentation, different routings and various firewall policies. [26]

Cloud load balancing provides routing for your application with great scaling and high availability options. Private or internet-facing applications can take advantage of the load balancing this service provides, either on the network level or on the HTTP(S) level. The network load balancing lets you distribute your traffic based on incoming protocol data, addresses and ports. HTTP(S) allows you to distribute the traffic across all the regions spread

around the globe so you can for example ensure that the traffic is routed to the closest one.[27]

Cloud DNS enables you to publish and maintain DNS records on Google's infrastructure. The service itself can be accessed also through RESTful API. It also provides great scalability and availability. The SLA promises 100% uptime on the authoritative name servers. Together with DNS forwarding and private zone, this service can be a great choice for making a hybrid solution and routing your requests into your on-prem solution. [27]

Cloud CDN creates a globally distributed content delivery network through a collection of edge points of presence which cache the HTTP(S) content in the locations closest to the end-user and thus dramatically increase the data transfer speeds. [26]

1.3.2 Computing

Cloud Compute Engine is the central service for Google Cloud Platform when it comes to provisioning, starting, connecting and overall management of Virtual Machine instances. The service enables you to create these Virtual Machines with different parameters as CPU, GPU, RAM, connectivity, base disk size or operating system image.[27] GCP also offers saving plans where you can commit for using the given resources for specified amount of time with a discount. Even bigger discount comes with preemptible instances of the Cloud Compute Engine. Preemptible instances can be terminated by GCP when they run out of spare resources and thus need the resources for their own services. [29]

Containers are an essential part of almost every architecture nowadays. GCP allows you to run containers either on the VM instances in Google Compute Engine by using container-optimized operating system images[30] or in Google Kubernetes Engine which provides container orchestration and can manage containers in a very scalable way and can even connect to your on-premises instances and create a hybrid solution. [31]

GCP also offers a serverless service called Google Cloud Functions. This service enables you to manage your code and takes care of the rest. There is no need to provision any Virtual Machine instances or container resources in any way.[1] GCP's free tier includes 2 million of free invocations per month and is billed per second based on CPU and RAM resources allocated to the task after the free tier amount is exceeded. [32]

1.3.3 Databases

When it comes to databases in Google Cloud Platform, the main service here is the Cloud SQL. It offers managed instances of MySQL, PostgreSQL and SQL servers. This service provides an option to also run these databases in a highly available environment where they are replicated to other zones and regions to ensure the maximum uptime. The service comes with automatic backups, storage increases and scalability if the requirements for the running environment grow over time. [33]

For NoSQL solutions, GCP offers multiple services such as Cloud Bigtable for key value store or Firestore for document structured databases. Cloud Bigtable is a fully managed cloud native database offered by GCP for largely scaled workloads which require low latency. The second NoSQL database, Firestore, is also a fully managed and serverless database offered by GCP. It offers rich support for a lot of programming languages not only from the mobile development environment in form of client-side and server-side libraries.[34]

1.3.4 Storage

Cloud Storage is GCP's object storage service available for companies with various sizes. The service offers essentially unlimited storage and no minimum size of the objects you store, low latency, high durability and geo-redundancy, so the data is stored in multiple regions to ensure availability and security if the physical location would be compromised. It also includes a range of storage classes which you can use to optimize the storage of your data and get better billing options because of that. There is Standard Storage, which is used primarily for frequently accessed data, Nearline Storage is a low-cost option which is the most suitable for storing data for at least 30 days, Cold Storage used for data stored for at least 90 days, Archive Storage is class with lowest cost and is great for storing long term data you would like to archive for at least 365 days. [35]

Unlike the other providers, the GCP does not offer any specialized service for disk storage. The disk storage is integrated directly into the Cloud Compute Engine and is commonly tied to the VM instances. There are zonal persistent disks, regional persistent disks, local SSDs, cloud storage buckets, firestore. [36]

1.3.5 Others

GCP's biggest advantage is Kubernetes for sure as Google is the original creator. Other most beneficial service would probably be Google Anthos, a service for advanced hybrid cloud environments. If we consider other services, GCP offers a lot of Artificial Intelligence, Machine Learning and Internet of Things services. GCP also allows you to pay for a complete SaaS solution when it comes to a platform for meetings with their Meet service, Google Workspace, formerly known as GSuite, as a solution for mailing needs and for example Google's Maps Platform for using Google's map APIs. [37]

1.3.6 Pricing

GCP's pricing model is very competitive as they are offering 300 dollars' worth of credits usable for all the services they provide. Google Cloud Platform also offers a Cloud Pricing Calculators so you can make estimates before you commit to actually using GCP's services. [13]

1.4 Cloud computing options comparison

Comparing modern state of cloud computing world is a very complex task because most of the big companies in the field already filled the gap which was previously present in this sector. Amazon Web Services is undeniably the biggest of the three big tech companies (Amazon Web Services, Microsoft Azure, Google Cloud Platform), mainly because of the head start and number of services they can offer to potential customers, but as mentioned previously, the gap is definitely closing as the time is passing by. [38] Microsoft Azure and Google Cloud Platform are concentrating a lot of resources to open-source software and solutions. Microsoft Azure is dedicating their focus towards their services like Windows Servers, System Center and Active Directory while Google, as the creator of Kubernetes, is focusing on containerization and container orchestration using their Google Kubernetes Engine. Regarding pricing, there is no clear winner either as all the cloud providers are continuously pushing their prices downwards to gain competitive advantage over the others.[40]

On-Demand Pricing						
Instance Type	AWS	Azure	Google	AWS pricing (per hour)	Azure Pricing (per hour)	Google pricing (per hour)
General purpose	m6g.xlarge	B4MS	e2-standard-4	\$0.154	\$0.166	\$0.156
Compute optimized	c6g.xlarge	F4s v2	c2-standard-4	\$0.136	\$0.169	\$0.235
Memory optimized	r6g.xlarge	E4a v4	m1-ultramem-40	\$0.202	\$0.252	\$6.303
Accelerated computing	p2.xlarge	NC4as T4 v3	a2-highcpu-1g	\$0.90	\$0.526	\$3.839




Figure 6 Comparison of VM instance pricing as of February 1st 2021[40]

II. ANALYSIS

2 MIGRATION TO THE CLOUD

A journey to the cloud involves enormous amount of research, planning and resource management. Whole migration process should be definitely divided into different phases to reduce complexity and possible blast radius. Some of the first thoughts when moving into the cloud should be the driving factor which will drive the whole migration process. There are definitely numerous factors and some of them might be more relevant than others. Some of these can include developer productivity, global expansion, standardized architecture or a data center lease expiry. One of the driving factors for this migration is for example not enough personnel specialized in physical infrastructure management and as the cloud providers provide a lot of managed services or at least services where you do not have to manage any physical infrastructure and you only have to provision virtual resources which you just simply attach to other virtual resources like virtual machines or different cloud services specialized for provisioning virtual server instances. Because of these facts this is one of the driving factors for BrandMaster. Migration can also be a great time to wipe out any cobwebs in the back of a closet. As the cloud providers offer a lot of SaaS services it can be beneficial to replace old and legacy solutions for the new ones.

The migration process also involves a lot of knowledge and overview over the whole application portfolio. Knowing all the interconnections and dependencies might be crucial later on during the migration process when you get into planning out the migration phases. Knowing what to migrate first and how to migrate is one of the most important information. The complexity of migration for different archetypes of applications may vary as different migration strategies offer different solutions. For example, containerized applications are going to be fairly easy to migrate with minimal changes needed and on the other hand big monolithic applications will be on the high complexity spectrum.[4]

The best generally considered migration approach is continuous improvement, starting from the least complex applications or services and learning from actually migrating non-complex applications. This way all the teams responsible for the migration will get more fluent and confident in the cloud environment. There might also be a need for running a parallel environment in the cloud and on-premises. This may bring doubled expenses and it may be necessary to complete the migration as soon as possible to mitigate these expenses. That might for example require you to dedicate certain amount of quality assertion resources to

validate and test the migrated applications and services in real time to minimize the time required to proceed with the migration and move to the next phase. [41]

2.1 Migration strategies

Migration strategies depend on the chosen architecture and licensing of application or services. While in the research of the application portfolio phase, it's very useful to figure out what types of applications are in your environment and what are you actually planning to migrate into the cloud, what is going to be easy and what is going to be hard to migrate.

Every application goes through a decision stages which ultimately lead to a production use of that given application. We often search for a general template we can really use on all the things we need. The template for migration to the cloud is often referred to as "The 6 R's" which describe the 6 most common application migration strategies when moving infrastructure and application to the cloud. The name comes from the processes this template describes. It includes Retention, Retirement, Rehosting, Re-platforming, Repurchasing and Refactoring. This process starts at a discovery in the application portfolio we are trying to migrate over to the cloud and ends at a validation stage, where usually the QA department validates migrated components or services and confirms the migrated component works correctly. After validation comes Transition. The application or service is transitioned into use in the cloud provider's environment and the on-premises solution is slowly turned off. When that happens, we enter the Production stage where we run the application solely in the cloud and we operate only within the boundaries of the given solution. Everything in between is a path we must carefully choose as each of them is beneficial for different scenarios. The first two, Retirement and Retention, are the easiest ones to execute as they do not require any action or require very minimal amount of action in order to work properly.

Retention is as simple as leaving migrated application or service behind on the on-premises servers. Sometimes not moving the application is an option which should be considered. Hybrid solutions are definitely right solutions when the situation demands it. [42]

Retirement is also a very simple path we can take during a migration. It essentially means moving the application or a service out of the active use and decommissioning it. This strategy should be considered when thinking about old, legacy solutions written long time ago or which we cannot actively maintain. [42]

Rehosting is also known as “lift and shift” and it provides two separate branches where we can choose between migrating using automation and manual migration where the process of installing, configuring and deploying is manually replicated inside the cloud environment. This path should be considered when migrating huge legacy applications or services. This migration can be fairly quick when opting for the automated tools which are provided for a lot of solutions for virtualized environments like VMWare where you can essentially just export and import a VM using specialized images created directly in the VMWare environment. The manual way of migration is also a very valid option as the team migrating the application or service manually can learn about underlying cloud components used in the automated migration and also learn more about the legacy application if the knowledge is not already spread among the team responsible for the migration. The manual path involves replicating the whole process of operating the application in the cloud environment. That involves the building, configuration and deployment of the migrated application. This strategy does not account for too much optimization or re-architecture as the only thing we do is simply moving the application by lifting it and shifting it to the cloud environment as the name suggests. [42]

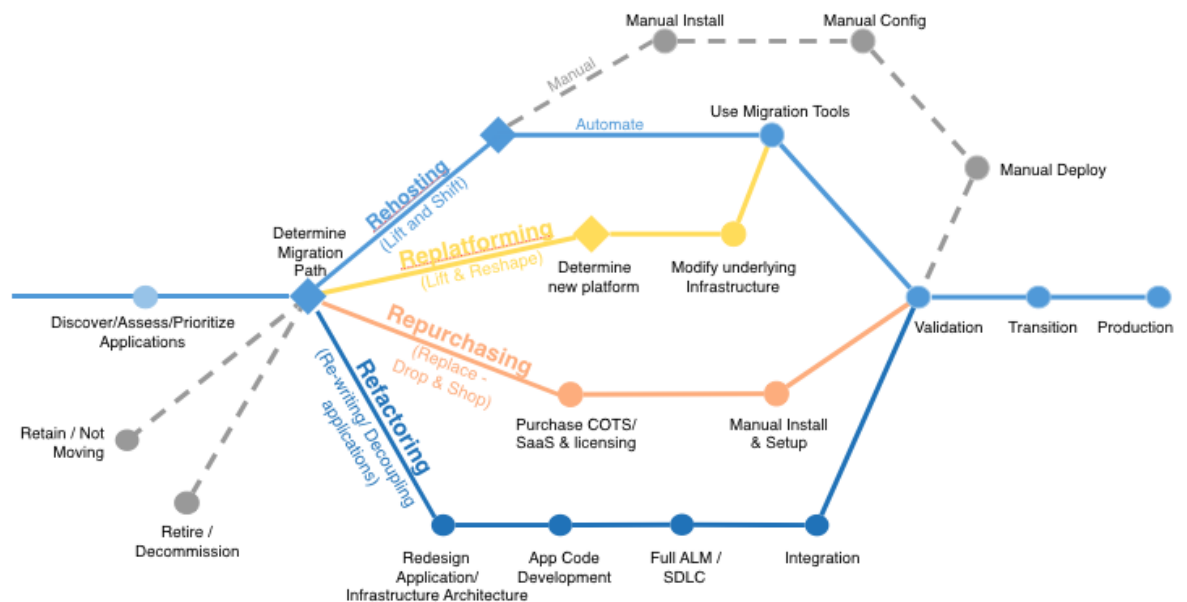


Figure 7 Six Rs of migration transformed into a graph [42]

Re-platforming is sometimes called “lift tinker and shift”. This strategy has a similar name to the rehosting for a reason and as the name suggests there is some tinkering involved before we finally move the application or service to the cloud. The tinkering is based on taking advantage of the cloud provider’s services or infrastructure, but not changing the core

architecture of the migrated application. The actual steps described in the strategy are about finding the new desired platform and evaluating its benefits and ultimately modifying the migrated application so it can work with the new cloud components. One of the possible use-cases of re-platforming could be moving from an on-premises hosted database towards a fully managed database service like Amazon RDS or using Amazon Elastic Beanstalk to migrate applications to a fully managed service. Other re-platforming examples might be moving away from old legacy or enterprise solutions and moving to a new and maybe cheaper or completely free solution. [42]

Repurchasing is as simple as moving to a new product. The most common repurchasing strategy is a move to a Software as a Service platform. A lot of cloud providers are offering a diverse range of Software as a Service products from emailing solutions to messaging platforms and developer tools. [42]

Refactoring also called Re-architecting or Re-imagining is the most complex strategy to migrate as it involves completely rewriting, re-architecting and redeveloping the whole architecture of the application or even the application itself. Typically, the goal is to use cloud native features and components as they are usually provided through an API or a library depending on the platform or a language the application is written in. Common use-cases might be transferring a large monolithic application into microservices, cloud functions or other totally serverless components. This strategy brings the most opportunities to add features, scaling, performance and generally tune the application at its core which would be expensive or quite difficult to do otherwise. [42]

2.2 Going into the cloud

The important factor during the migration is to stay up and running for the whole time without any disturbances to the delivery of the services to the end-users and customers which can be achieved through a hybrid solution for the duration of the migration process. One of the driving factors is not enough qualified staff which would be able to manage the physical infrastructure on-site. The other important driving factor is pricing or expenses needed to maintain the infrastructure. With cloud providers this need is eliminated by default as the majority of products which are being offered are managed by the cloud providers themselves. These managed services also usually provide better pricing even though the developers do not have to manage any physical infrastructure while using the managed services. The major advantage of cloud computing is the pay as you go billing model which

saves up major amount of money as all the resources do not have to run all the time. During interruptible or short-lived workloads, the infrastructure is able to automatically scale and downscale dynamically and save computing time that way in comparison to self-hosted environment where it would be impossible or very hard as this setup would have to be configured manually whereas in the cloud environment this setup is automatically available upon registering an account.

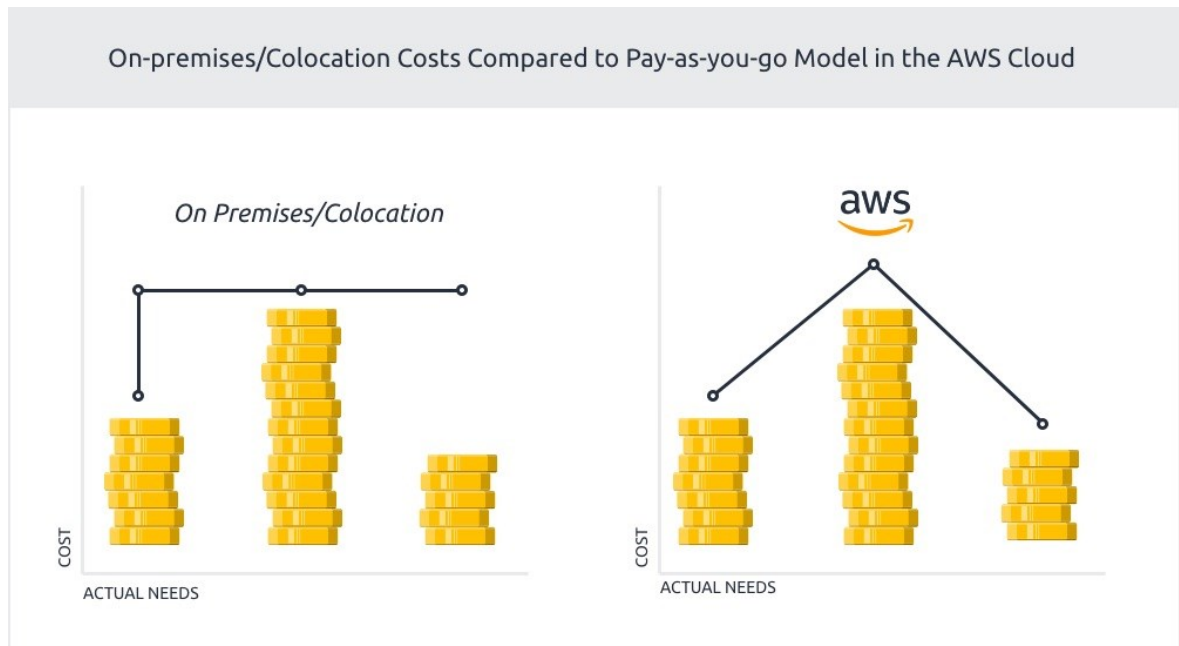


Figure 8 On-premises costs compared to pay as you go billing model[43]

The chosen platform for this migration is Amazon Web Services. The AWS is the major big tech company on the market which is leading with the number of managed services and number of components which can be used to build the solution from ground up overall. The BrandMaster company already has a competence in hosting parts of its solution in AWS so it is only logical to choose the cloud provider the development team is already at least a bit familiar with. AWS also makes their services available in the desired regions and they also make improvements to critical components BrandMaster solution needs to use. The example of this is including additional transitions between tiers in S3 buckets as the data inside the solution is very diversly spread among the data coldness and hotness axis. This function will provide great statistics and save money on expenses as the data will move into its proper tiers after the specified time to make the transitions between pricing tiers.

2.3 BrandMaster application portfolio and architecture overview

Brandmaster’s current infrastructure and application portfolio consists of a few essential parts. There is a very diverse mix of applications from legacy components running on Delphi or JBoss server hosted on bare servers to totally newly created microservices running inside Kubernetes using newest standards and Java Spring framework. From the point of view of programming languages, the portfolio is split into Java applications, Kotlin applications, Delphi applications, PHP applications, Nodejs applications, Front-end applications written mostly in Angular and huge set of APIs running inside an Oracle database using PSQL procedures. In terms of databases, the portfolio includes a huge Oracle database which consists approximately half of the data stored among all the databases, then there is a PostgreSQL database used primarily for Java/Kotlin applications and MySQL database used mainly for PHP applications.

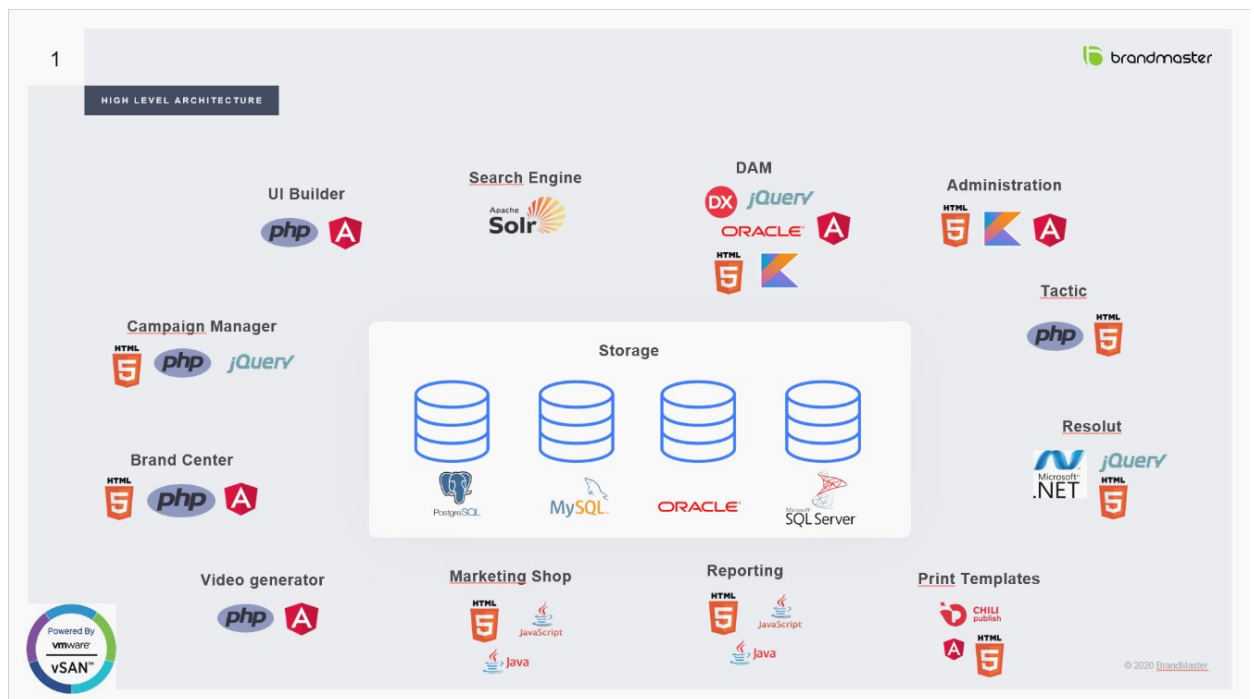


Figure 9 High level technology overview of applications and connected databases

We can also divide our applications into architectural groups. There would be a group of monolithically oriented applications and the ones which are more microservice oriented. The last separation can be done on the technological level. Java backend services mostly use Tomcat, either one monolithic server, where classical Java Servlets are being deployed, or in the embedded form when it comes to the Spring and microservices. Even though the

Tomcat server takes the majority of the whole java infrastructure there are also services which are running on a Netty server and are using reactive programming principals.

PHP applications, together with the Oracle database PLSQL APIs, are running behind an Apache server. They are routed through load balancer and a front web server consisting of multiple Nginx servers. When we take the path from the other side, the traffic arrives at the Oslo datacenter where it is spread among the front load balancers which send the traffic to the Nginx servers directly behind it from where the traffic is routed to the individual load balancers dedicated to specific applications or to a group of specific applications like applications running inside a Kubernetes cluster. All the infrastructure in the whole BrandMaster ecosystem was being run on virtualized servers in a VMWare vSphere environment.

2.4 Legacy applications

In BrandMaster's infrastructure legacy application mainly include a huge Oracle database instance with variety of PLSQL packages and procedures, old Java application running on a Jboss server and a range of Delphi services running on windows servers.

The heart of the whole system is the enormous Oracle instance which essentially holds over 50% of the data the BrandMaster's ecosystem. The long-term plan is to decommission this huge instance, mainly because of maintenance issues, but it is included in the migration as it has proven difficult and very time-consuming process to refactor the functionality it provides to newly architected micro services written either in Kotlin or Java using the Spring framework. The database runs on a bare, virtualized Linux based server which requires scheduled restarts because of memory leaks associated with the connections towards the PLSQL procedures. The database mainly holds data from the Java legacy application running on the JBoss server. Other than that, the database is the source for most of the data in the Digital Asset Management (DAM) application. It provides authentication and authorization throughout the whole system, stores all user data, all the privileges and interconnections for most of the BrandMaster applications. The other most important thing it holds is a lot of Single Sign-In functionality made specifically for certain clients. Basically, every application is connected to the Oracle database either directly or indirectly, so this component is crucial until its most essential functionality is refactored to newer services. Probably the most important application running in Oracle would be Mars Portals which is a search index of all companies and clients in the BrandMaster's ecosystem. It lists all the

applications and settings which are set for the given company and allows administrators to move across companies, log into them and set a large variety of settings, privileges and flags on different levels of the system.

The second most important application is a legacy Jboss application written in Java. It is maintained primarily because it holds a component called Legacy Login. It's a login to a Legacy system which mainly used to provide Flash based visual editors for printing and advertising templates. As the Flash reached its end-of-life support BrandMaster is working on moving last clients away from this solution. Nevertheless, the Legacy login is still important for other components as it still provides the login capability for a lot of users and clients.

The third and very important part of the legacy infrastructure are Delphi services. These services are running on multiple virtualized Windows machines. The services provide an old API towards our Solr search index which is used primarily to search through assets uploaded into the DAM archives. It also manages 80% of graphical conversions, asset creation, scaling, and mainly transcoding of all uploaded assets into the DAM application.

All of these applications are to be discontinued and decommissioned, but they are really tightly integrated into the system, so the removal of these services is impossible at this point in time, and they are planned to move with all the other components in the system into the new infrastructure inside AWS.

2.5 Modern applications

More modern applications are split into two big groups divided by a programming language they are programmed in. Java or Kotlin and PHP are two primary ecosystems used on as the server technologies. PHP ecosystem primarily consists of a Toolbar application, Campaign manager application, Brandcenter and Brandbook applications, UI builder, email editor and a sharing application. Java application ecosystem is quite larger than that when it comes to the number of services. It includes a Marketing Shop, Reporting module, Core module, Form Builder module, Vendor module, Dataset module, Print Advertising module, Asset Picker, Shorty, Data Warehouse service, Billing service, Queue service, Color management service, Template management service also called Chili service, JWT service, API gateway, DAM API, Solr API, Category API, Admin API, Marketing Shop API, Partner Portals, Template Groups, My Creatives and all back-end for front-end applications. There is also one other,

very important group of code and those are front-end applications for these back-end applications. They are mostly written in Angular framework, and some are in plain vanilla JavaScript.

If we start with the PHP ecosystem, then the Toolbar application is responsible for building and displaying a toolbar which is an essential element which is displayed pretty much everywhere throughout the whole system, on all pages and every view. Its primary intent is to navigate between modules, display logged-in user information, settings and account preferences. Campaign manager is a time planning application which provides a very modifiable timeline where user defined events can take place. It is primarily used to plan out marketing and advertising campaigns but can be also used to host internal events and presentations. The application also provides advanced functionality for form creation and distribution among the selected user base participating in the given campaign. Brandbook and Brandcenter are the two most visited applications as they are the front of most BrandMaster's clients. As the name suggests the application is used to gather information about the given brand and its display in a nice way to the end user. The main advantage of this application is that it has advanced WYSIWYG editor which lets you build the pages without any prior knowledge of a programming or a markup language. The editor consists of various predefined elements which the user can move around and style to his liking. For more advanced users there is also more functionality where the end user can style the elements himself and achieve more personalized looks and environments. Sharing application is a relatively new containerized and simple application used for sharing pages to social medias like Facebook, Twitter, Pinterest and others. Email editor, as the name suggests, is a simple email editor which lets you define email templates which are distributed among the specified portfolio of users. The UI builder is a big application which lets system administrators define custom styles and designs for all or a specified subset of applications in BrandMaster's portfolio. The styles can be defined in UI and then they are built and distributed to specific servers for the selected company or client. The architecture of the PHP applications is quite simple as they are running on a classical LAMP stack consisting of a Linux server or container, Apache web server which is routing the requests to the application itself, PHP applications which are primarily written in a Nette framework, and a MySQL database with a few connections to the legacy Oracle database.

Java application ecosystem consists of three types of applications or modules. First one being an application called BM2013, which will become legacy soon, running on multiple load

balanced Tomcat application servers, and a few services running on a Netty server. These applications are running on a bare virtualized server without any type of containerization. The other type of applications are applications written in Spring Framework or Quarkus and coded in Kotlin programming language. These applications are always containerized and specially optimized for running in a container runtime or they are even built as cloud native applications with special Java Virtual Machines.

The BM2013 applications is one fairly big Java application split into several smaller modules. The application uses Java Servlet API to provide REST API for more than 15 applications which are slowly being refactored to micro services written in Kotlin in Spring or Quarkus framework. Some of the core functionality of the modules include essential core functionality aggregated in the Core module, general API for e-shop functionality capable of tracking inventory for articles, assets or tickets provided by the Marketing Shop module, various statistics reports in the Reporting module and Print advertisement ordering through the Print Advertising module. Shorty is one of the smallest services in the whole ecosystem. It is used to shorten the URLs specified by the user. It runs on a Netty server, and it has exactly two endpoints. One for creating the shortened URL and the other one for using the generated URL. Next important service is an internally developed queuing system which basically integrates with almost all the visual editors within the BrandMaster's ecosystem and usually generates resources created in them asynchronously or initiates asynchronous procedures somewhere in the system. The newest subsection of the whole Java ecosystem is the Spring framework micro services written in Kotlin. Every new project which belongs in the Java section is created this way as the Spring framework brings a lot to the table and accelerates the development process thanks to a standard and custom-made starter packages which the team just need to configure. There are currently two ways to access these services and that is through HTTP request sent to a JSON API and listeners AMQP messages. These ways are of course depending on how the starter packages are configured. The overall architecture for these newer services is based on a front service which is called API Gateway. Every request needs to go through this service as the backends behind it require a JWT token to authenticate and authorize the user. There are security measures which guarantee correct redirects to client's login pages if the user is not authenticated, authorized or if the session is invalidated. When the API Gateway authenticates the user, the request gets routed to the desired service, if it exists, with the JWT token in the HTTP requests. The finishing service

authorizes the user and checks its privileges and eventually returns the result the request is asking for.

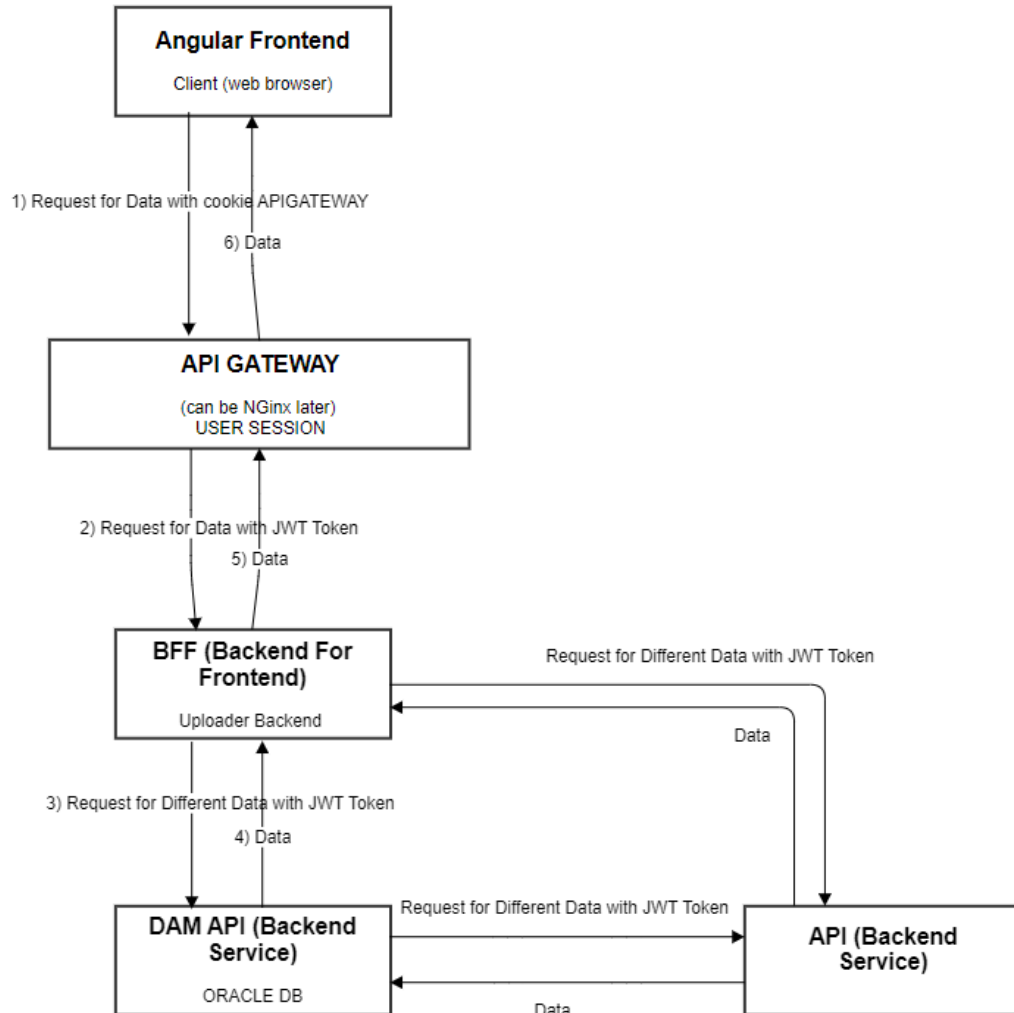


Figure 10 Authentication architecture for new Spring applications

2.6 Data

Probably most important, but not recognized part of each system is of course the storage. As the nature of the BrandMaster’s product the storage of the marketing data, printing and advertisement templates is certainly important. There are several clustered data servers, and their primary purpose is to store data. The size entire solution which falls under the migration is around 150 TB after an intense cleanup. This data is also distributed among different disks with different properties. Some of it is cold and some of it is hot so it is definitely important to differentiate between the nature of the data.

2.7 Other components in the infrastructure

There are of course a lot more components involved in providing the infrastructure which is needed to run BrandMaster's solution. The front of the whole system is a Nginx web and proxy server. It takes care of the routing of the requests towards the right back-end servers and applications, plus it is a front for almost over 150 custom domains BrandMaster is hosting for its clients. The Nginx server is running in a load balanced manner so there is additional redundancy for peak loads sent towards the system. Before entering the system, the requests go through Checkpoint firewall which ensures no dirty traffic gets into the system. After the Nginx jump, the variety of paths broadens quite a bit as there is a large range of servers the request can be routed to. Depending on if the application runs in a load balanced manner, it may be routed through a HA-Proxy server which is acting as the load balancer. The final piece to the routing puzzle is the internal DNS server running on bind. It essentially provides a way to call the services in the same way within different environments of the BrandMaster system.

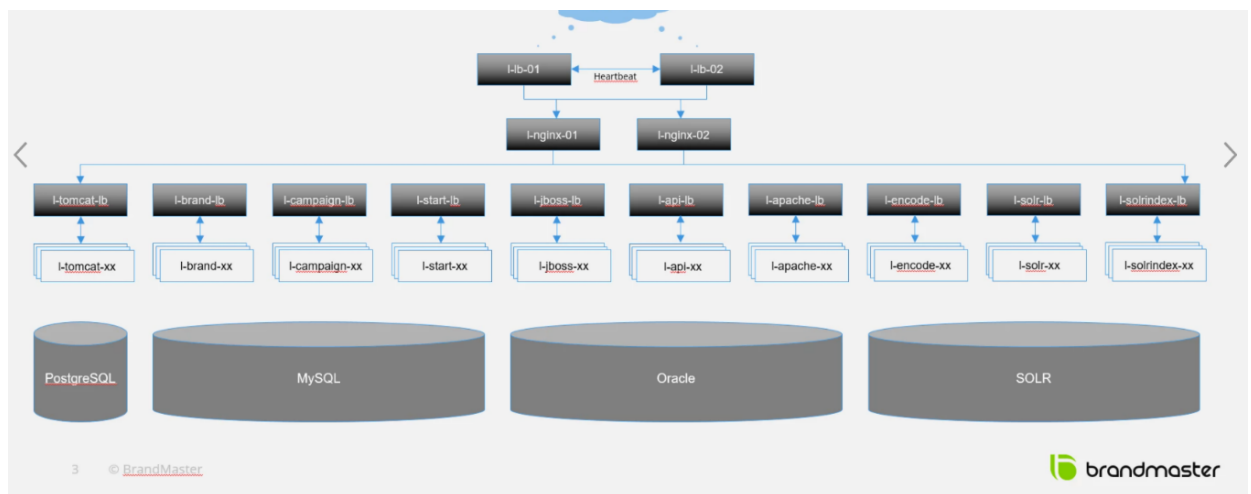


Figure 11 Illustration of the internal infrastructure

RabbitMQ is a messaging system BrandMaster chose to use as a modern solution to send asynchronous messages throughout the system. It is currently used for graphical resource generations, notifications and company cloning jobs.

Caching is an important component, especially when it comes to delivering content like graphical templates, pictures, printing materials which can reach extreme sizes. Two important services which are used are Redis and Memcached. Redis is primarily used in the Java ecosystem as it integrates flawlessly with the Spring framework with high-level libraries but also with low-level wrappers around its API. The Memcached is mainly used in

the PHP section as it is a simple key value pair storage which is perfectly suited for the use of the PHP applications.

Other essential component throughout the system and mainly in the DAM application is searching. The searching capabilities are mostly backed by the SOLR search engine. It runs in a sharded way to ensure data redundancy in case of failure. Because there is no authentication nor authorization in the SOLR APIs there is a micro service called SOLR API which basically just builds requests towards the SOLR APIs and return the results based on the level of access the user has.

Last but definitely not least critical component is logging. All the exceptions or errors from most applications in the ecosystems are sent to a Sentry service with notifications set up in individual groups of developers which should know about. Other part of the logging system is a Graylog server which sole purpose is log aggregation from the Kubernetes clusters and containerized services. The log aggregation is being done using multiple fluent bit instances or specialized side-car components which send the logs over to the Graylog server.

2.8 Application connections

The interconnections within the system are quite frequent as a lot of applications are based on some visual editor mostly used to create or create marketing materials for our clients. The heart of the system is the Oracle database as it holds most of the system data and the legacy solution written as PLSQL procedures. Most older applications are coupled quite tightly with the Oracle database. Some examples might be DAM and Legacy. The newer applications are developed and designed in such a way so the coupling is kept to a smallest amount possible or so there is no coupling at all and so they can be switched, refactored or replaced any point in time without any hiccups or technical problems. Most applications are also loosely coupled with Reporting application as its primary purpose is to gather and aggregate statistical data from all around the system and provide it in a nice readable way to the end-user. As one of the primary goals of BrandMaster is to provide visual editors for marketing and printing materials to its end-users there are naturally a lot of editors which are sometimes very tightly integrated with different services. For example, if we take a look into the DAM application. It can store every template technology provided and supported by the BrandMaster's ecosystem so DAM must be able to either view preview from these visual editors or even provide to edit the materials directly in the DAM's interface and generate new materials. The graphical editors are also tied to a lot of data supplying services like Data

Sets where the end-user can either define or provide data in various formats and create a data set BrandMaster applications can then iterate over these data sets and create batch jobs to create multiple cards with different names and faces, leaflets with randomized texts and images, really anything the end-user specifies. Next integration might be Marketing Shop where the created materials can be offered to the different branches of customers. Other example of loosely coupled application might be My Creatives. The service basically keeps track of the user’s graphical asset generation jobs, and it reports back with basic progress status. Again, all the BrandMaster’s template technologies are integrated so the application essentially gets notification and messages from the entire system in a generic way, so it can easily be integrated into newly created services and newly integrated or supported template technologies.

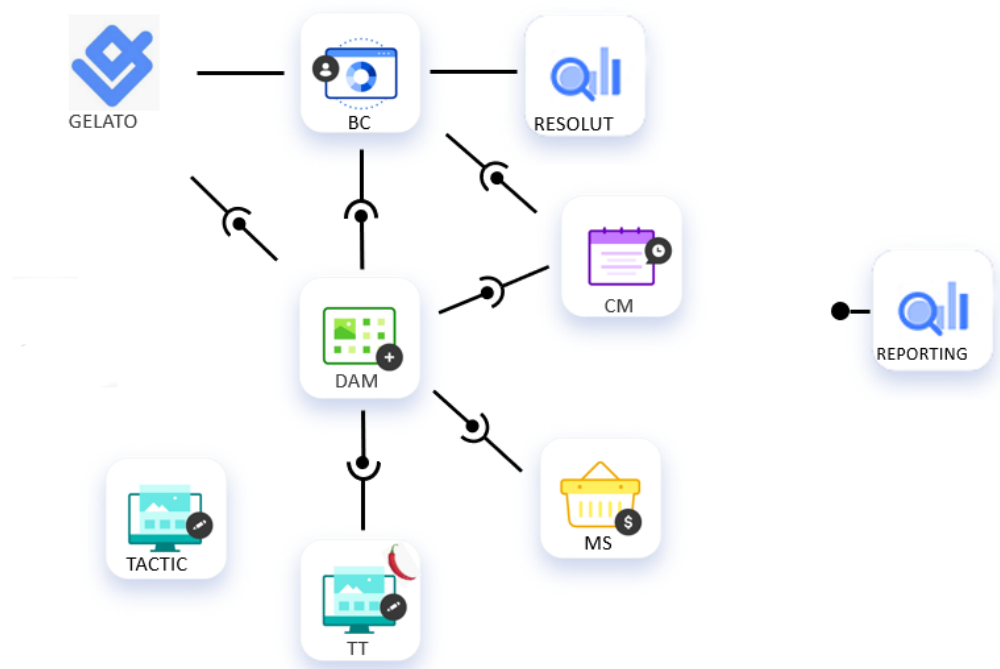


Figure 12 High level application connections including external connections.

3 MIGRATION PLAN

One of the most important decisions to make during a migration is the migration plan. It should consist of every step you are planning to make during the whole transition period including selected technologies or substitution for the selected technologies, transition steps with possible rollback plan, separation into phases where several types or sections of on-premises software or infrastructure is going to be moved and so on. When planning for the Brandmaster's infrastructure migration there were several points considered. Some of the considered things were client consents we needed in order to migrate their solution to the cloud, double expenses which are created by the fact that the on-premises solution and cloud solution must coexist for a certain timeframe during the migration and rollback steps needed to be taken to ensure the stability of the migration if some problems arise. Because of these double expenses the decision was made to make the migration about the speed of the migration and not about unnecessary optimizations. There are also various design decisions to be made when migrating to the cloud. For example, one of the design decisions for this migration was to script the entire infrastructure solution using Infrastructure as Code. It allows the team to create the entire infrastructure in readable YAML configuration files which are used together with AWS CloudFormation service. This service takes the specified templates and create the described infrastructure automatically according to specified parameters. There are also different ways to distribute these parameters, for example using Parameter Store or Secret Manager. The templates can be interconnected and can pass exported variables from the created infrastructure resources between each other. Together it makes a powerful system which can create entire solutions or systems from the network layers up to application deployments in minutes. This fits into the BrandMaster's use case very well because of some potential use-cases where clients demanded a dedicated solution. Another decision made was the fact that the two solutions are going to coexist for some period of time during which they must be interconnected and create a hybrid solution which will be able to connect resources from one solution to another and vice versa. This decision was made based on the migration strategy which was split between three phases where essential components of the system are going to be migrated. The last important thing was to get consents from the clients themselves. The nature of the cloud solutions is to be distributed, serverless architecture is becoming increasingly popular in the cloud world but it can be particularly important for your clients to know where you store their data, how are you planning to protect it, what vendors are you going to use and what certifications they

have. The collection of these consents has proven to be the most time-consuming phase of this migration as BrandMaster is hosting the solution for more than hundreds of clients which consist of numerous subunits. So, it is certainly important to dedicate enough time for this phase of the migration. The migration was divided into three phases which separate the infrastructure and how it is migrated. First wave was dedicated to traffic, routing and essential infrastructure and primarily training of the team in working with the AWS ecosystem and infrastructure. Second wave was reserved for migration of data, databases and selected applications and the third one consists of moving every other component which is not tightly coupled to anything in one big bang.

3.1 First phase

The decision was that in the first phase of the migration the primary subject will be to move all the network traffic coming into the on-premises solution, so it goes through AWS. By moving to AWS and making it front for the whole solution, you create an environment where you can essentially route requests however you want, mainly meaning you can route to AWS and already migrated services running on the AWS infrastructure, and you can also route to the on-prem solution and still use all the services without any unwanted disturbances. According to plan, the front for the whole solution was provided by AWS CloudFront which is primarily a service for creating HTTP endpoints in a distributed way all over the world. The service has rich cache capabilities, it can create TLS connections in connection with certificates managed by AWS Certificate Manager. Its primary purpose is to be the front for the whole solution and host the custom domains for every client BrandMaster has and also route the HTTP requests to the underlying resources deeper in the system like S3 or load balancers which can route the requests on lower levels depending on application needs. The second but not less important goal of this phase was to create all the essential infrastructure which is needed to migrate everything from the on-premises solution to the cloud. Including AWS VPC, subnet separation, different routings on the network level, DNS and connections towards the on-premises solution through VPN tunneling.

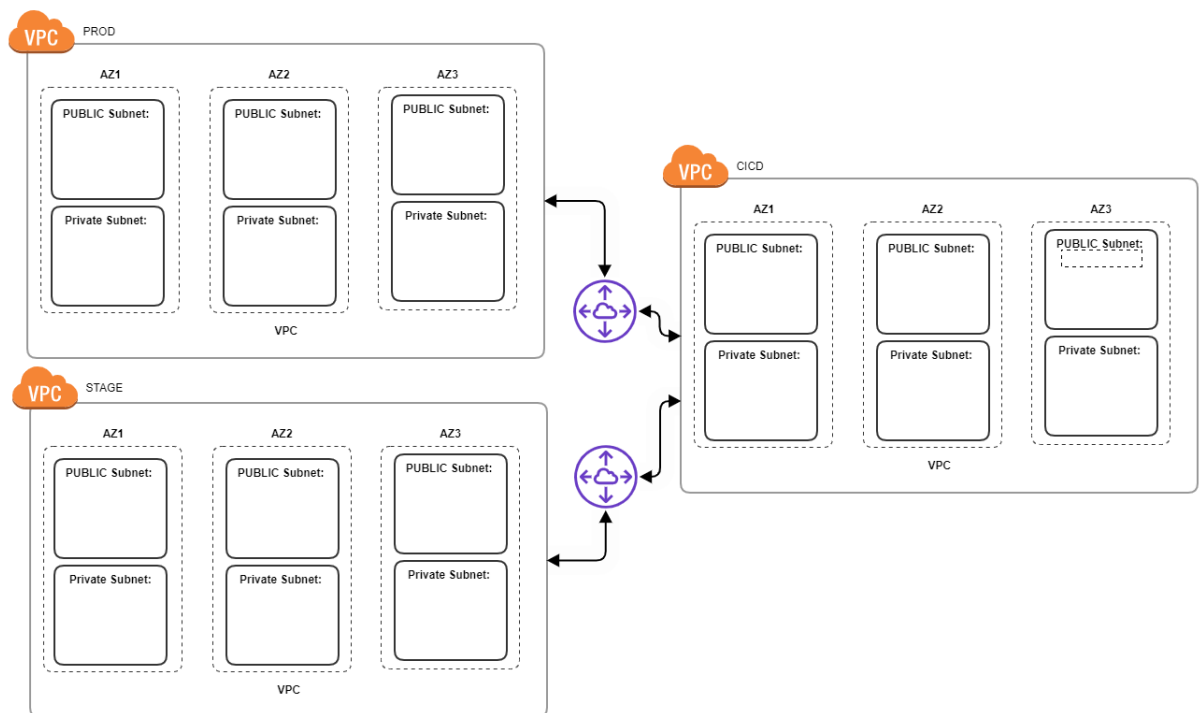


Figure 13 VPC connections between different development accounts

First two simultaneous steps in this phase are the essential infrastructure and getting rid of technical dept which built up over time by setting up older clients with direct A records instead of a CNAME records which could be moved to a different one during the process of this migration. This is a process which can be done simultaneously while preparing the essential infrastructure like account separation, VPC, subnet segmentation and connecting the network to the on-premises solution through VPN tunneling.

The account structure heavily depends on what suits your needs. BrandMaster’s account structure is separated into the development environments together with accounts added to mimic AWS structuring.[5] The accounts include a development account, stage account, production account for running applications, a CI/CD account which is able to connect to the application accounts and is primarily used to run build and deployment and tools which need to access each of the environments. Two more accounts exist specifically for the needs of AWS infrastructure. Logging and IAM accounts are separate accounts for those needs. Logging account is primarily used for storing logs from the whole system and all the accounts. The function of the account is to provide these logs to all developers in a read only fashion so they can debug the applications but cannot alter or delete them. The IAM account is a special account which is essentially hosting all the users used for authentication into all the other accounts in the BrandMaster ecosystem. AWS function called “Role assumption”

is a function where a user can assume the given role which is setup in a way so it can access resources from different accounts. Using this role assumption and a proper two-way trust relationship between the given accounts result in access to permitted resources through IAM policies for the selected and assumed role. This is how developers can authenticate towards different accounts throughout the BrandMaster accounts.

The essential infrastructure is composed of multiple building blocks starting with a Virtual Private Cloud, which basically is a separated network which is dedicated solely for your needs. By creating a Virtual Private Cloud, you create a space where you can create subnets and their segmentation according to your specification and needs. The architecture chosen for BrandMaster includes running in three availability zones which ensure the maximum uptime of the whole network in the chosen region. Each availability zone has a public subnet, which is connected to an internet gateway which connects the subnet directly to the internet, and a private subnet which is used to run all the application which are connecting to the internet through a NAT gateway. The whole network is interconnected with the on-premises network using a VPN gateway. The VPN gateway has two connections which include two tunnels to ensure the maximum uptime if some of the tunnels would go down. The final building block of the essential infrastructure is Route 53 which is a very scalable DNS server as a service provided by Amazon. The service is used to forward DNS requests if the server get requests for IP addresses under the on-premises network the requests get forwarded. The same happens on the on-premises network and DNS servers towards the AWS network. This is how the two systems are interconnected through specialized domain names dedicated towards this connection during the migration process.

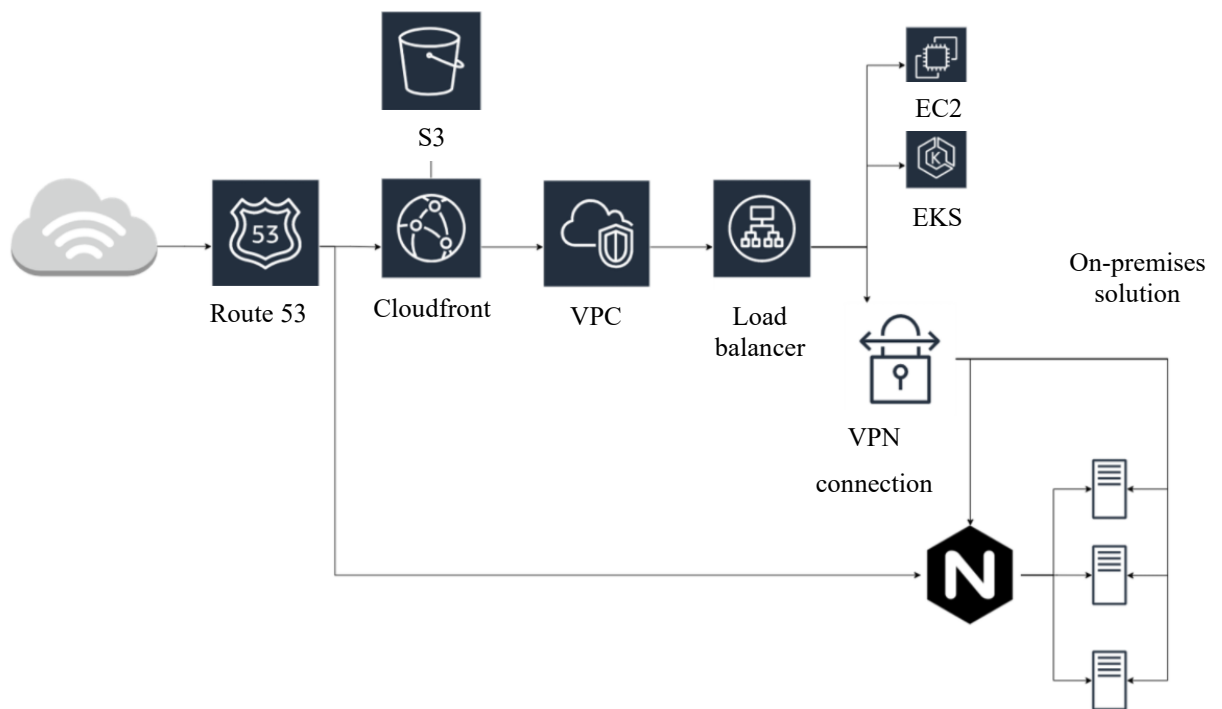


Figure 14 Routing architecture which keeps the cloud solution connected with the on-premises solution.

The second step of the first migration phase is moving the certificates for client domains into AWS and its Certificate Manager service from which the CloudFront distributions can get the certificates to use them to provide HTTPS connections on the given domains. There is a restriction for CloudFront distributions for certificates. A CloudFront distribution can hold only one certificate at a given time, so the solution is to spin up a distribution for each certificate, meaning for one client, which is hosted separately in the BrandMaster's solution. This spin up is done via the CloudFormation script in an Infrastructure as Code way and it is fully automated using the YAML templates and bash scripts. When a request gets to the CloudFront it gets evaluated by the Web Application Firewall service in AWS and as it iterates over the ruleset configured in the service it either gets blocked or allowed and is let pass through to the load balancer which is sitting behind the firewall. The load balancer has its own rule-based routing according to the requested path which is routed to the correct back-end service. Depending on the weighted DNS, set in the Route 53 service, the request gets routed either to a service running in the AWS environment or through the VPN tunnels to a service running on the on-premises network. Thanks to all of this, the services can run without any disturbances even during the migration of any given service or part of the infrastructure as everything is routed through the weighted system of the Route 53 service and until the DNS records are switched to point towards the AWS infrastructure setup the

requests and all the traffic is routed to the on-premises solution through the VPN tunnels connected to it.

3.2 Second phase

The beginning of the second phase was about moving the data together with the databases in the BrandMaster's ecosystem. The main data storage service in AWS ecosystem is S3. It is an object storage service which basically saves the objects on a path which can imitate a folder structure. There was definitely one major problem when deciding to use this service as the main source of the storage for the whole system and it was the fact that the BrandMaster solution is dependent on mounting the storage server folders as disk shares as NFS shares on the physical server the services are running on or through Kubernetes persistent volumes. This is not possible to do by default and design as S3 does not have that kind of interface exposed to the outside world by default. This would require rewriting every service in the BrandMaster ecosystem to use an AWS library so it can communicate with the S3 over HTTP and store the files that way. There is however a service called a Storage Gateway. This service is used to create a hybrid storage, but it can also create NFS volumes from the S3 buckets and that was very convenient for BrandMaster as there would not have to be any rewrites done during this migration and it saved a lot of time which would have to be spent on these rewrites. The Databases were migrated over to AWS using one of the database specific tools depending on if the migrated database was Oracle, MySQL or Postgres. They were migrated to the Relation Database Service, shortly RDS, which provides the managed instances for each of these databases. Furthermore, the MySQL and the Postgres databases were migrated to the special AWS Aurora instances to integrate the databases with the AWS infrastructure and so they have the support for all the advanced maintenance features AWS can offer for these specialized instances. There was an initial try to move the databases through the automated tools and through a cluster setup which would simply replicate the data but during the process the team encountered several issues with replicating important indexes and sequences which would not retain its state after the replication and would reset so the resolution for the problem was to spend a bit more time on scripting an automated way which would manually dump the necessary data and import it into the final RDS instance in AWS. This solution would create a downtime which had to be accounted for during the migration as the synchronization of the data through this process can take up to a few hours depending on the delta between the data which would be

synchronized periodically and then the data which would be migrated during the downtime of the database during the migration.

The initial testing of the first phase showed promising results thanks to the caching the CloudFront is providing but during the second phase the system response times got abnormally slower in the core parts of the system connected primarily to the Oracle and Postgres database. This was discovered in the later stage of the second phase and the decision was to expand the scope of this phase to include some of the core applications as the response times were not satisfying enough to keep them as they were.

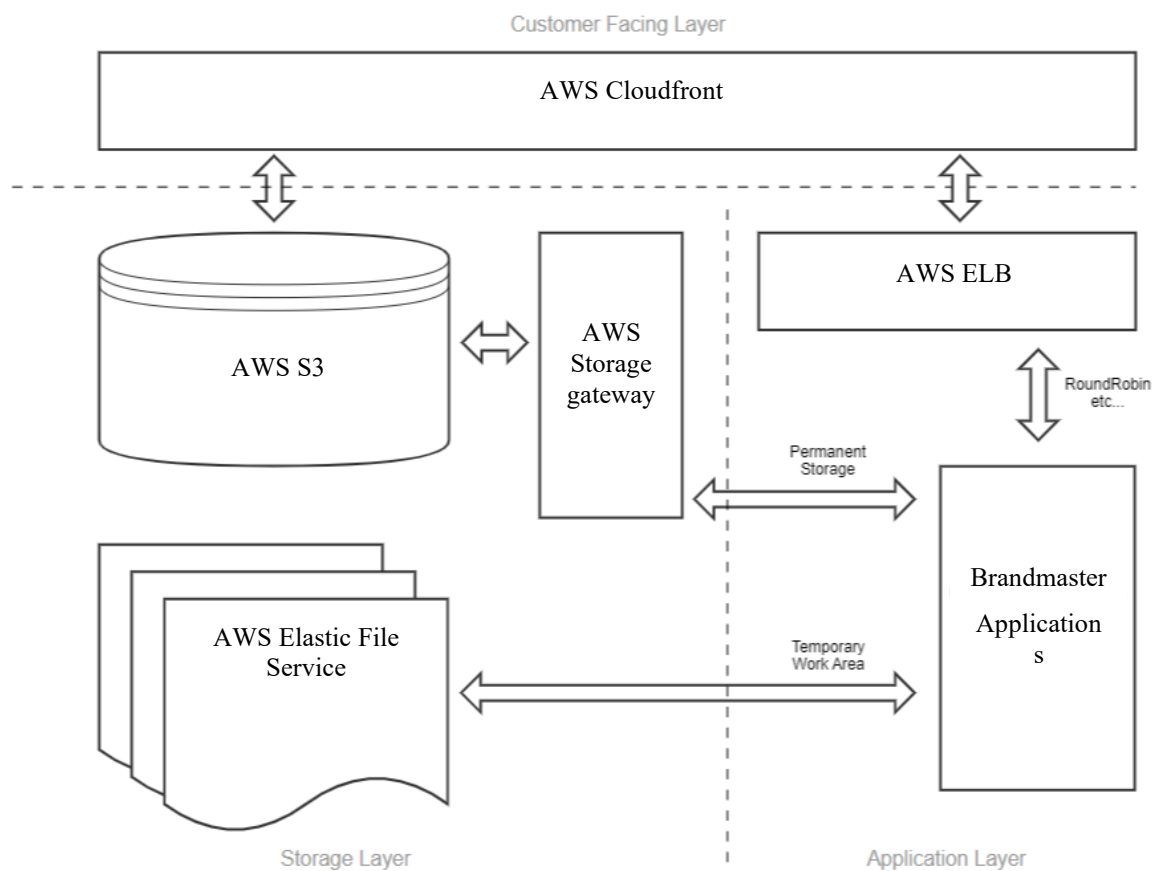


Figure 15 Storage Architecture for the AWS solution

The amount of data moved over to the AWS surpassed 150 TB mark. After the initial investigation into the data migration services the final decision was to use S3 sync to synchronize the data over to AWS. The data was synchronized over the normal open Internet as moving the data through the VPN tunnels could throttle the bandwidth and potentially cripple the whole connection between the environments and the price of the migration would increase drastically as the VPN tunneling is charged based on the traffic moved through the tunnels themselves. The option to move the data over the internet is only limited by the on-

premises bandwidth limitations and there are no additional charges included in moving the data over to S3 as there are no charges on the incoming data transfers.

The two simultaneous steps during this second phase of the migration process were to establish a running CI/CD pipeline which would be equivalent to a bamboo server setup running on the on-premises solution at the give date and setting up a Kubernetes cluster in the AWS infrastructure as a preparation for the third phase which would migrate over all the services including the Kubernetes cluster running on the on-premises solution. After thorough look over the portfolio of the AWS services offered, the decision was to opt for the Atlassian Bit Bucket Pipelines solution as it provided exactly the same functionality the bamboo setup provides to this date. The main reason for opting for the Atlassian solution was because AWS does not support any dashboards for viewing what patches of the applications are currently running on the target environments at any given time, there is also no comparison when deploying the patches. Atlassian provides these features right out of the box, so the decision was to go with the Bit Bucket Pipelines as they are replacing the Bamboo server functionality according to the Atlassian cloud roadmap.[44]

3.3 Third phase

Third and last phase was focused on a big bang where all the application and services were migrated in fast consecutive waves. There were several lift-and-shift applications which mainly included the old, legacy, and obsolete solutions which are going to be out of service in some specified time frame. The easier part of the third phase was the migration of all containerized applications as they do not require any major changes and they can be moved pretty much as they are. The only difference was in the CI/CD pipeline which had to be set up for special AWS infrastructure and all the connection had to be set up towards the AWS. This phase is important because this is the time when the on-premises solution was shut down completely and all contracts terminated as all the infrastructure and services were moved to the AWS infrastructure.

4 APPLICATION MIGRATION AND DEPLOYMENT

The application migration was divided into two parts, migrating the legacy and old applications in a lift-and-shift manner by manually exporting the VM images from VMWare and manually importing them into the AWS environment and migrating containerized applications just by adjusting the deployment procedure.

The legacy applications were migrated one by one in a manual fashion. The prior solution was running in a virtualized environment in VMWare vSphere which was running all the virtual machines, connected the data clusters and provided environment for a lot of integrated containers. The migration steps mainly include exporting the image from the VMWare vSphere environment and importing it into the EC2 AMIs. By exporting the image from VMWare environment, we are able to do the lift-and-shift migration. The image transfers the state of the virtual machine, and it can also migrate the state of all the disks attached to the virtual machine. The building of these application has got changed because of the nature of the migration. The new build of these applications newly includes the need to build the virtual machine AMI image where the internal structure of the application, configuration files or other states is modified during the build using the EC2 image builder. The image is then deployed on the EC2 computing engine either in a load balanced way or as one virtual machine instance based on the nature of the given application.

The containerized applications were migrated in a big bang fashion, meaning the applications were moved almost all at once. The final migration was divided into smaller waves separated by a small-time margin to test the application sections properly. The building of the application themselves do not change but the platform the applications are being built on changed from the Bamboo server to the Bit Bucket Pipelines as Atlassian is moving away from the self-hosted and cloud instances of Bamboo server. All the Java applications are being built using a Jib utility made by google which creates optimized Docker and OCI images with layered jars inside them. Normally the images are being built with a fat jar, meaning the container contains a jar as a normal physical or virtualized server would. The layered jar strategy is taking advantage of docker layer caching. Each layer of the layered jar is stored in one of the docker layers. It is usually split into more static libraries which do not change frequently, user-defined libraries which usually are third-party libraries that change frequently to keep them up to date due to security reasons and to the application code itself which is the most frequently changed layer. Images created this way take less

time to upload and download from repositories and also save a lot of data and startup time costs in the cloud. PHP applications are being built as a standard Docker images through builds specified in the Dockerfile included in the root directory of each application. These images are collected in image repositories in the AWS Elastic Container Registry. The images are then deployed in the same way using the kubectl CLI utility which is standardly used to communicate with the target Kubernetes clusters. Kubernetes uses special YAML specification files which define what Kubernetes components should be deployed and how they should behave. One component which is being deployed is a Service which basically aggregates all the pods of one application and load balances incoming traffic between them. The second component is the pods, which is the lowest unit of the Kubernetes ecosystem, and it encapsulates the containers running in the runtime environment for the containers. The Kubernetes pods are being deployed using rolling updates, meaning the pods are being gradually updated and only as the services running inside pass a specified health-check, only then they are transitioned into the currently running and all the incoming traffic is being routed to the new pod instances.

5 TESTING OF THE MIGRATED APPLICATIONS

The solution was always tested during all the phases of the migration. One of the reasons to move to a cloud solution was to make the content more globally available and more accessible all around the globe with faster response times and download speeds. Of course, there were preparations for some response times increase during the migration period where there would be some kind of a hybrid solution which connect the AWS cloud solution and the on-premises solution. The expectations also were that the content is immediately going to be more available as the CloudFront distributions are able to cache the specified paths on the edge locations spread around the world and as the clients are naturally closer to these edge locations the expected download times of files are expected to be lower.

The first phase was tested on response times from the initial on-premises solution and then compared with the response times from the solution moved to the CloudFront distributions with caching enabled on the edge locations. The testing brought very expected results. The initial on-premises testing resulted in a standard time for the whole solution which is periodically tested normally. When the CloudFront solution was tested, the result showed an expected increase in response times for back-end API HTTP calls by approximately 23% caused by the traffic going a longer way though the VPN tunnels to the on-premises solution and a great decrease in times for downloading files by approximately 60% caused by the caching mechanism, which is caching the files closer to the clients resulting in this great decrease of times while downloading the given files.

	Final Execution 1			Final Execution 2		
	First Launch	Second Launch	Difference	First Launch	Second Launch	Difference
Application Call (average in ms)	658	762	-13,6%	640	968	-33%
File Download (average in ms)	1234	772	+59,8%	1177	741	+58,8%
Total Average	791	765	+3,3%	764	915	+16,5%

Figure 16 Results of the initial environment and CloudFront testing.

The worst performance decrease came with the migration of the databases to the AWS RDS instances. All of the older and naturally slower solutions and applications are tightly coupled

to the bigger databases from the BrandMaster ecosystem. The performance degradation came from the fact that the databases got moved even further from the applications themselves and the traffic had to move through the VPN tunnels which resulted in a significant increase in response times. The resulted times increased even by several seconds which was clearly unacceptable. This performance degradation was corrected by a later decision to move the tightly coupled applications to the AWS ecosystem sooner in the second phase with the databases rather than later during the big bang when all the applications were moved. After this additional scope was added to the second phase the response times stabilized and even improved by a small margin of 5%.

The last phase removed the remaining performance decrease caused by the traffic going through the VPN tunnels to the on-premises solution. By moving the applications to the AWS infrastructure, the system response times unified as all the parts of the solution were moved to AWS in a wave fashion. There were some more increases in performance as other rules for caching were set up and the durability of the AWS infrastructure showed minor improvements in response times between the services which communicated over the internal AWS network.

6 MONTHLY EXPENSES OVERVIEW AND COMPARISON

The initial on-premises solution was hosted in a datacenter located in Oslo in Norway. The datacenter required an initial investment, and that fact naturally increased the expenses in first year when putting together the physical infrastructure for the BrandMaster development and hosting environment. The price needed to maintain the infrastructure stabilized from the second year until the present time and the approximate final price for maintaining the infrastructure per month is \$47,510 when the expenses, according to internal reports, for the whole year are divided into twelve months. The expenses include stable operating cost of all the storage, computing and traffic which goes through the datacenter. They also include rack space and power; high speed internet access and it also includes multiple licensing deals together with hardware replacement and maintenance. Major items on the list which can be definitely saved upon are the operating costs for computing and storage as today these two are either up and running all the time or is decided manually by the application and it is fairly difficult to migrate to other storage classes based on the data coldness or hotness. These can be decreased by using the managed services like EC2, S3 or RDS. They provide functionality for lower price without the need of managing the physical infrastructure with the additional benefit of paying only for the resources the solution actually uses.

New platform cost	Y0	Y1	Y2	Y3	Y4	Y5
<i>Operating cost - storage</i>		732 000	732 000	732 000	732 000	732 000
<i>Operating cost - compute</i>		1 040 000	1 040 000	1 040 000	1 040 000	1 040 000
<i>Operating cost - traffic</i>		137 000	137 000	137 000	137 000	137 000
<i>Implementing costs - external</i>		-	-			
Existing platform - Total HW, SW and operating cost	-	1 909 000	1 909 000	1 909 000	1 909 000	1 909 000
<i>Datacenter cost - Rackspace and power</i>		298 800	298 800	298 800	298 800	298 800
<i>Datacenter cost - Internet to DC</i>		96 000	96 000	96 000	96 000	96 000
<i>HW investments - replacements outdated HW</i>		500 000	500 000	500 000	500 000	500 000
<i>SW and HW - Storage and compute (vmware)</i>		439 000	439 000	439 000	439 000	439 000
<i>SW - Maintenance - HP3PAR (CJH)</i>		55 000				
<i>SW - Oracle reinvestment for Standard license</i>		363 000	80 000	80 000	80 000	80 000
<i>SW - Microsoft License</i>		550 000	550 000	550 000	550 000	550 000
<i>Oracle Operations parter cost</i>		400 000	400 000	400 000	400 000	400 000
Total changes in Operating costs - Existing platform	-	2 701 800	2 363 800	2 363 800	2 363 800	2 363 800
Total Changes/Implications		792 800	454 800	454 800	454 800	454 800

Figure 17 Expenses for the on-premises solution for building the business case in BrandMaster (Norwegian crowns)

The main advantage of the cloud computing is in using only the resources the solution actually needs as it is very easy to provision other resources in times of peak traffic or other

extreme situations. This also includes automatic scaling of the given services during these situations. As one of the main driving factors of this migration was pricing of hardware the solution does not actually use and the lack of educated staff in managing the physical infrastructure on site. The usage of managed services and cost-effective offerings by AWS the monthly expenses lowered to approximate of \$22,459, when adding all the expenses from all the environments and accounts for the time we hosted the solution in the cloud. This price tag consists of all the accounts not counting in the development environment as it was decided to not always use this environment after this migration anymore. The environment should only be spun up only if there is a serious need for it. If the final price would be split into percentages connected to each environment or account in this matter the majority of the expenses would naturally come from the production account as more powerful computing resources are used there and majority of the data fall under the production account. The stage account would be the second when it comes to expenses as this environment is basically a mirror of the production environment with less powerful computing resources and data stored only for testing purposes for relevant customers. The third is the CI/CD account where all the tooling lives. The least expensive accounts would be the IAM and logging account as the IAMs are integrated into the core of AWS system and the costs are minimal and the logging account only generates costs based on the storage used which can be controlled with log retention policies.

Moving into the cloud also includes a lot of autoscaling and on-demand expenses in times when the solution experience significant traffic and load spikes. This can make the price fluctuate quite a bit as the spun up resources naturally increase the specified expenses. This also works in the other way around. In times when the solution does not need the specified resources it is very easy to downscale the operation and save up on those expenses on holidays and other occasions when the traffic and load towards the solution is lower than usual. The major item on the billing list will always definitely be the cloud computing services as the nature of all solutions is to compute. The cloud solutions usually have better and more efficient ways to compute the desired tasks and get rid of the used resources after it finishes its tasks. This way the price can be significantly decreased by rewriting and decommissioning the old solutions which would be too hard to containerize or rewrite during the migration and it would hold it back for much longer time.

CONCLUSION

The migration the cloud is definitely a time-consuming process which require a great deal of planning and preparations in before the actual migration plan is set to motion. Migrating also requires a lot of knowledge about the target platform so it is better to plan with a certain training period for the responsible personnel, so they can educate themselves and there can be some preparation work done like in the case of the BrandMaster migration - getting rid of technical dept or cleaning up or separating the data which is going to be migrated. There are already specified routes to migrate to the cloud and BrandMaster application definitely fall into some of those specified routes and categories. It is very effective to follow the guidelines specified by these routes as it makes the transition faster and quite efficient, but it can be very beneficial to create some brand new specially tailored solutions for the infrastructure you are migrating because there usually is at least one part of the infrastructure which is going to require a tailored migration strategy which will suit your needs.

The plan to migrate can be beneficial in regard to modernizing and securing the operations of the migrated solution. There also can be specific requirements which need to be met as for this migration there was the need to interconnect the solutions and keep all the services up and running during the whole process of migration.

Testing is one of the most important processes to do during a migration as there can be some unforeseen side-effects during a migration in a hybrid-like setup as this one. The minimal response time increases we accounted for, but manual user testing unveiled a significant increase in the second phase of the migration when the databases got moved and the core legacy applications slowed significantly because of the traffic being routed through the VPN tunnels to the AWS and back. So, it is important to setup routines and a baseline on the initial environment so the results can be compared and interpreted for the most accurate and best results overall.

This migration clearly shows that migrating to the cloud can greatly decrease the operation costs if the solution is either container oriented and the costs could be decreased even more if the infrastructure and all the components would be refactored into cloud native variants as these services provide the best cloud integration such as fast start up times and integration with other cloud services out of the box. These costs can be decreased furthermore by deprecating certain components and decommissioning them after a period needed for them to be rewritten into a cloud native form.

BIBLIOGRAPHY

- [1] SULLIVAN, Dan, 2020. *Official Google Professional Cloud Architect Study Guide* [online]. Indianapolis: Wiley [cit. 2020-10-19]. ISBN 978-1-119-60244-6. Available at: <https://www.wiley.com/en-us/search?pq=1119596572%7Crelevance>
- [2] SAVILL, John, 2020. *Microsoft Azure® Infrastructure Services for Architects: Designing Cloud Solutions* [online]. Indianapolis: Wiley [cit. 2020-10-19]. ISBN 978-1-11959657-8. Available at: <https://www.wiley.com/en-us/search?pq=1119596572%7Crelevance>
- [3] PIPER, Ben a David CLINTON, 2019. *AWS Certified Cloud Practitioner Study Guide CLF-C01 Exam* [online]. Indianapolis: Wiley [cit. 2020-10-19]. ISBN 978-1-119-49070-8. Available at: <https://www.wiley.com/en-us/search?pq=1119490707%7Crelevance>
- [4] PIPER, Ben a David CLINTON, 2019. *AWS Certified Solutions Architect Study Guide* [online]. Second Edition. Indianapolis: Wiley [cit. 2020-10-19]. ISBN 978-1-119-50421-4. Available at: <https://www.wiley.com/en-us/search?pq=111950421X%7Crelevance>
- [5] PERROTT, Sara a Brett MCLAUGHLIN, 2020. *AWS Certified SysOps Administrator Study Guide* [online]. Second Edition. Indianapolis: Wiley [cit. 2020-10-19]. ISBN 978-1-119-56155-2. Available at: <https://www.wiley.com/en-us/search?pq=1119561558%7Crelevance>
- [6] CLOUD COMPUTING: What are cloud service providers?, 2021. *Redhat.com* [online]. Raleigh, USA: Red Hat [cit. 2021-4-28]. Available at: <https://www.redhat.com/en/topics/cloud-computing/what-are-cloud-providers>
- [7] RICHTER, Felix, 2021. CLOUD INFRASTRUCTURE MARKET: Amazon Leads \$130-Billion Cloud Market. *Statista.com* [online]. Hamburg: Felix Richter [cit. 2021-4-28]. Available at: <https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/>
- [8] Co je cloud computing?: Průvodce pro začátečníky, 2021. *Microsoft Azure* [online]. Redmond, Washington, USA: Microsoft [cit. 2021-4-28]. Available at: <https://azure.microsoft.com/cs-cz/overview/what-is-cloud-computing/>

- [9] VENNAM, Sai, 2020. Cloud Computing. *IBM* [online]. Armonk, New York, USA: Sai Vennam [cit. 2021-4-28]. Available at: <https://www.ibm.com/cloud/learn/cloud-computing>
- [10] Overview of Amazon Web Services: AWS Whitepaper, 2021. *Amazon Web Services* [online]. Seattle, Washington, USA: Amazon Web Services [cit. 2021-4-28]. Available at: <https://d1.awsstatic.com/whitepapers/aws-overview.pdf>
- [11] AWS Lambda changes duration billing granularity from 100ms down to 1ms, c2021. *Amazon Web Services* [online]. Seattle, Washington, USA: Amazon Web Services [cit. 2021-4-28]. Available at: <https://aws.amazon.com/about-aws/whats-new/2020/12/aws-lambda-changes-duration-billing-granularity-from-100ms-to-1ms/>
- [12] Amazon Web Services Review, c2019. *PC Mag* [online]. New York, USA: Wayne Rash [cit. 2021-4-28]. Available at: <https://www.pcmag.com/reviews/amazon-web-services>
- [13] SOLANKI, Jignesh, c2021. Cloud Pricing Comparison 2021: AWS vs Azure vs Google Cloud. *SIMFORM* [online]. Austin: Jignesh Solanki [cit. 2021-4-28]. Available at: <https://www.simform.com/compute-pricing-comparison-aws-azure-googlecloud/>
- [14] Co je Azure?, c2021. *Microsoft Azure* [online]. Redmond, Washington, USA: Microsoft [cit. 2021-4-28]. Available at: <https://azure.microsoft.com/cs-cz/overview/what-is-azure/>
- [15] Microsoft Azure Portal, c2021. *Microsoft Azure* [online]. Redmond, Washington, USA: Microsoft [cit. 2021-4-28]. Available at: <https://azure.microsoft.com/cs-cz/features/azure-portal/>
- [16] Azure networking services overview, c2021. *Microsoft Docs* [online]. Redmond, Washington, USA: Microsoft [cit. 2021-4-28]. Available at: <https://docs.microsoft.com/en-us/azure/networking/fundamentals/networking-overview>
- [17] Přehled výpočetních služeb Azure, c2021. *Microsoft Docs* [online]. Redmond, Washington, USA: Microsoft [cit. 2021-4-28]. Available at: <https://docs.microsoft.com/cs-cz/learn/modules/azure-compute-fundamentals/overview>

- [18] Supported languages in Azure Functions, c2021. *Microsoft Docs* [online]. Redmond, Washington, USA: Microsoft [cit. 2021-4-28]. Available at: <https://docs.microsoft.com/en-us/azure/azure-functions/supported-languages>
- [19] Azure Functions pricing, c2021. *Microsoft Azure* [online]. Redmond, Washington, USA: Microsoft [cit. 2021-4-28]. Available at: <https://azure.microsoft.com/en-us/pricing/details/functions/>
- [20] Azure managed databases, c2021. *Microsoft Azure* [online]. Redmond, Washington, USA: Microsoft [cit. 2021-4-28]. Available at: <https://azure.microsoft.com/en-us/solutions/databases/>
- [21] Seznámení se základními Azure Storage službami, c2021. *Microsoft Docs* [online]. Redmond, Washington, USA: Microsoft [cit. 2021-4-28]. Available at: <https://docs.microsoft.com/cs-cz/azure/storage/common/storage-introduction>
- [22] Introducing Google App Engine + our new blog, c2021. *Google App Engine Blog* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://googleappengine.blogspot.com/2008/04/introducing-google-app-engine-our-new.html>
- [23] App Engine 1.6.0 Out of Preview Release, c2021. *Google App Engine Blog* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://googleappengine.blogspot.com/2011/11/app-engine-160-out-of-preview-release.html>
- [24] GCP Dashboard Overview, c2021. *Medium* [online]. San Francisco: Medium [cit. 2021-4-28]. Available at: <https://medium.com/google-cloud/gcp-dashboard-overview-c80ffd0ed521>
- [25] Google Cloud Platform Services Summary, c2021. *Google Cloud* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/terms/services>
- [26] Networking services, c2021. *Google Cloud* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/docs/overview/cloud-platform-services#networking>
- [27] MP, Alexis, c2021. GCP Essentials: GCP Compute Platform Overview. *Medium* [online]. San Francisco: Alexis MP [cit. 2021-4-28]. Available at:

<https://medium.com/google-cloud/new-gcp-essentials-episode-gcp-compute-platform-overview-cc73d583d9d0>

- [28] Concepts, c2021. *Google Cloud* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/compute/docs/concepts>
- [29] Compute Engine, c2021. *Google Cloud* [online]. Mountain View, California, United States.: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/compute>
- [30] Containers on Compute Engine, c2021. *Google Cloud* [online]. Mountain View, California, United States.: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/compute/docs/containers>
- [31] Google Kubernetes Engine, c2021. *Google Cloud* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/kubernetes-engine>
- [32] Cloud Functions, c2021. *Google Cloud* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/functions>
- [33] Google Cloud databases, c2021. *Google Cloud* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/products/databases>
- [34] Cloud SQL, c2021. *Google Cloud* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/sql>
- [35] Cloud Storage, c2021. *Google Cloud* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/storage#section-10>
- [36] Storage options, c2021. *Google Cloud* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/compute/docs/disks>
- [37] Google Cloud products, c2021. *Google Cloud* [online]. Mountain View, California, United States: Google [cit. 2021-4-28]. Available at: <https://cloud.google.com/products>

- [38] CAREY, Scott, c2020. AWS vs Azure vs Google Cloud: What's the best cloud platform for enterprise? *Computer World* [online]. Needham: Scott Carey [cit. 2021-4-28]. Available at: <https://www.computerworld.com/article/3429365/aws-vs-azure-vs-google-whats-the-best-cloud-platform-for-enterprise.html?page=2>
- [39] AWS vs Azure vs Google: Detailed Cloud Comparison, c2021. *IntelliPaat* [online]. IntelliPaat [cit. 2021-4-28]. Available at: <https://intellipaat.com/blog/aws-vs-azure-vs-google-cloud/>
- [40] Cloud Pricing Comparison 2021: AWS vs Azure vs Google Cloud, c2021. *SIMFORM* [online]. Austin: SIMFORM [cit. 2021-4-28]. Available at: <https://www.simform.com/compute-pricing-comparison-aws-azure-googlecloud/>
- [41] A Process for Mass Migrations to the Cloud, c2021. *Amazon Web Services* [online]. Seattle, Washington, USA: Amazon Web Services [cit. 2021-4-28]. Available at: <https://aws.amazon.com/blogs/enterprise-strategy/214-2/>
- [42] 6 Strategies for Migrating Applications to the Cloud, c2021. *Amazon Web Services* [online]. Seattle, Washington, USA: Amazon Web Services [cit. 2021-4-28]. Available at: <https://aws.amazon.com/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/>
- [43] How to Control Your IT Costs in the AWS Cloud?, c2021. Heptabit [online]. Zagreb: Heptabit [cit. 2021-4-28]. Available at: <https://www.heptabit.at/blog/how-to-control-your-it-costs-in-the-aws-cloud>
- [44] Moving to a cloud future, together, c2021. *Atlassian* [online]. Sydney: Atlassian [cit. 2021-4-28]. Available at: <https://www.atlassian.com/migration/journey-to-cloud>

LIST OF ABBREVIATIONS

AWS	Amazon Web Services
GCP	Google Cloud Platform
EC2	Elastic Compute engine on the AWS platform
S3	Simple Storage service on the AWS platform
RDS	Relation Database Service on the AWS platform
HTTP	Hypertext Transfer Protocol
API	Application Programmable Interface
CLI	Command Line Interface
CI/CD	Continuous Integration and Continuous Delivery
OCI	Open Container Initiative
VPC	Virtual Private Cloud
VPN	Virtual Private Network
REST	Representational State Transfer
DAM	Digital Asset Management
PHP	Personal Home Pages
LAMP	Linux, Apache, MySQL, PHP a technological stack used to develop websites
JWT	JSON Web Token
PLSQL	Procedural Language for SQL
CDN	Content Delivery Network
IP	Internet Protocol
SLA	Service Level Agreement
AI	Artificial Intelligence
UDP	User Datagram Protocol
TLS	Transport Layer Security

TCP Transmission Control Protocol

QA Quality Assertion

LIST OF FIGURES

Figure 1 Market share of the leading cloud infrastructure service providers [7]	10
Figure 2 Illustration of management needs in different levels of infrastructure abstraction [9].....	12
Figure 3 Example of the AWS interface. [12]	13
Figure 4 Example of the Microsoft Azure interface[15]	17
Figure 5 Example of the Google Cloud Platform dashboard [24].....	21
Figure 6 Comparison of VM instance pricing as of February 1 st 2021[40].....	25
Figure 7 Six Rs of migration transformed into a graph [42]	29
Figure 8 On-premises costs compared to pay as you go billing model[43]	31
Figure 9 High level technology overview of applications and connected databases..	32
Figure 10 Authentication architecture for new Spring applications	37
Figure 11 Illustration of the internal infrastructure	38
Figure 12 High level application connections including external connections.	40
Figure 13 VPC connections between different development accounts	43
Figure 14 Routing architecture which keeps the cloud solution connected with the on-premises solution.....	45
Figure 15 Storage Architecture for the AWS solution	47
Figure 16 Results of the initial environment and CloudFront testing.....	51
Figure 17 Expenses for the on-premises solution for building the business case in BrandMaster (Norwegian crowns)	53