Overview of Cloud Computing

Michael Wufka, Ph.D.
CSIS Department
Douglas College
Canada

Massimo Canonico, Ph.D.
Computer Science Institute
University of Piemonte Orientale
Italy

For comments/suggestions:
wufkam@douglascollege.ca, massimo.canonico@uniupo.it

Version 1.3

The book was produced with Overleaf (https://www.overleaf.com).

Michael Wufka

Michael Wufka is a regular instructor in Computing Studies and Information Systems at Douglas College in Metro Vancouver, Canada. Prior to earning his PhD in Information Systems and Business Administration at the University of British Columbia, he completed his undergraduate degree in Computer Science at the University of Technology in Darmstadt, Germany, and worked for Accenture in a range of different roles on several IT projects for four years. His main research interests include cloud computing and agile software development. He is a member of the Google Cloud Faculty Experts Group since 2020.

Massimo Canonico

Massimo Canonico is an assistant professor of Computer Science at the University of Piemonte Orientale (Italy). His research interests lie in different areas, Concerning the Distributed Computing systems, his current research activity is focused on cloud computing with emphasis on performance and energy efficient resource management. He has also worked on fault-tolerant scheduling algorithms and efficient overlay-based data transfer techniques on various families of distributed systems (in particular, Grid Computing systems). Since 2020, he has been selected as a member of the Google Cloud Faculty Expert Group and as AWS Educate Cloud Ambassador by Amazon.

Foreword

Cloud computing in its modern form arguably started in 2006, when Amazon began to offer public cloud computing resources. One of the consequences of this rather young age is that significant innovations are still occurring at a rapid pace. In turn, this means that books on the subject tend to become obsolete quickly.

Universities started to propose courses related to cloud computing in the middle of the 2010s when the popularity of this platform was already high. Unfortunately, at that time, it was not easy to find a good textbook on the subject. As a matter of fact, most introductory books that offer an overview of cloud computing to a technical audience were either of a low quality or already significantly outdated. And this observation still remains true to the best of our knowledge. It seems that the size and complexity of the topic has led most authors to focus on specific areas of cloud computing. For example, there are great books on topics such as cloud security, how to program cloud native applications, or specific tools such as Kubernetes. For an introductory college course in cloud computing this is a challenge, as it is unreasonable to ask students to buy and read a number of expensive textbooks for a single course.

This book aims to fill the void described above - to serve as an introductory textbook on cloud computing. The target audience are readers with some technical background that are also interested in the business aspects of cloud computing. The book intentionally does not focus on technical details and does not include step-by-step instructions in order to avoid becoming obsolete too quickly. While new tools and concepts are sure to continue to come up at a rapid pace, the bulk of the book should remain true and useful for a number of years. Examples are usually based on the Google Cloud Platform, but the principles covered in the book are equally relevant to users of other cloud platforms.

Hands-on tutorials and other materials for learning cloud computing can be found at `https://tcc.uniupo.it`.

Acknowledgements

Writing this book mostly during a global pandemic was both rewarding and challenging. I could not have completed this work without the ongoing support and encouragement of my wife Karina and son Andreas.

Michael

I have to say thanks to my wife Manu, my son Jacques and my dog Mia, all of them supported me or asked me for a walk. I really appreciate both aspects during this year since most of my free time was dedicated to this book.

Massimo

We thank the "Google Cloud Faculty Experts group" which brought us together and allowed to start this useful collaboration. We want also thank all the people who provide us valuable feedback and suggestions to improve the content of the book. In particular, we thank Jacopo Mauro, Barry Redman and Stefan Wufka.

Massimo and Michael

Contents

Chapter 1

Fundamentals

This chapter gives a broad overview of cloud computing. The definition of basic concepts and terms is followed by a discussion of advantages and disadvantages of cloud computing. A brief history of computing paradigms from mainframe computers to modern cloud computing concludes the chapter.

1.1 What is cloud computing?

The *National Institute of Standards of Technology* (NIST) of the U.S. Department of Commerce defines cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."[1] This rather dense definition contains a number of important characteristics of cloud computing:

- **Ubiquitous [...] network access:** Cloud computing is accessed over the Internet. This makes it accessible wherever Internet connectivity is available.
- **On demand:** Cloud computing is inherently on demand - i.e., you are not required to plan ahead how much of a resource you will need at a specific time in the future. Instead, you can use as much of each resource as you spontaneously need (as long as you can pay for it), and you are only billed for how much you actually use.

[1]Peter Mell, Timothy Grance: "The NIST Definition of cloud computing", NIST Special Publication 800-145

- **Shared pool of configurable computing resources:** In order to allow the previous point, cloud service providers have a large pool of computing resources (such as servers, storage, network capacity etc.) that is dynamically allocated to their customers on demand.

- **Can be rapidly provisioned and released:** Not only can resources be used and released dynamically - this can happen very rapidly (typically within a few seconds, or at most, minutes).

- **With minimal management effort or service provider interaction:** The allocation and release of resources happens in a self-serve manner, e.g. through a few mouse clicks in a web interface or through an API call, and does not require any manual actions by the provider (which is an important prerequisite for the 24/7 availability of rapid provisioning)

A few examples[2] might make this definition more easy to understand. To begin with private cloud use: If you own a smartphone and occasionally take photos on it, odds are that you are using storage in the cloud. Typically, photos taken are automatically uploaded into a cloud (e.g. by Google or Apple) so that they are not lost if you lose your phone, and so that you can save storage space on your phone by removing old photos that have already been synchronized in the cloud. Another computing resource that is commonly used is applications. It is increasingly common to not install all the software one uses on one's computer, but to instead use software in the cloud. This might be typical office software (e.g. Google Docs or Microsoft 365) or design, image processing etc. software (such as the Adobe Creative Cloud).

Using cloud based software applications instead of locally installed and managed software is also increasingly common in organizations - everything from simple accounting packages to complex *Enterprise Resource Planning* (ERP)[3] systems can now be rented in the cloud. Another typical example of cloud use by organizations is to replace physical servers with virtual ones hosted in the cloud.

All these examples have in common that users do not have to commit to how much of each resource they will use in advance. Instead, use is on demand, with additional resources (more storage, more copies of cloud software for additional users in the organization, additional servers etc.)

[2]More examples will be discussed in Section 1.2.1 on elasticity.

[3]More info here: `https://en.wikipedia.org/wiki/Enterprise_resource_planning`

available virtually instantaneously, without any interaction with service provider personnel. And users are only charged for the amount of resources they consume.

For each type of resource, the cloud service provider typically defines a relatively fine grained basic unit of use, and the cost associated with the use of that unit. For example, storing a GByte of data for a month might cost \$0.026, or running a virtual machine with 16 CPUs and 64 GByte of memory for an hour might cost \$0.53. The total charges for a cloud user are calculated simply by multiplying the number of units used of these (and many other[4]) resource types and their individual unit cost.

Cloud computing has different service and deployment models that are discussed in detail in the next two sections.

1.1.1 Cloud Service Models

Cloud services are typically categorized into three main types of service models: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) and *Software as a Service* (SaaS).

In **IaaS**, the cloud service provider offers virtual machines, networking and storage capacities. Everything on top of these bare resources (e.g. configuring and maintaining operating systems, applications etc.) is the responsibility of the cloud service user.

PaaS shifts more responsibilities from the cloud service user to the cloud service provider. In a certain sense PaaS is the most vague of the three service models (simply because not everybody exactly agrees on what a "platform" is). In practice, different cloud service providers offer different kinds of platforms at different levels of abstraction. To use a simple example here, a provider could define the *LAMP* stack as a platform. LAMP stands for Linux, Apache, MySQL and PHP, and consequently represents the combination of a very popular operating system, web server, database and programming language. There are many applications that can be directly deployed on the LAMP stack - the content management system Wordpress which is used by millions of blogs is a good example. In PaaS, the cloud service provider is responsible for maintaining the platform, and the cloud service user can focus on deploying and maintaining their application(s) on the basis of the platform.

[4]For many more examples of cloud resources, their basic units of use, and respective price, see e.g. the Google Cloud Platform price list: `https://cloud.google.com/pricing/list`

SaaS moves even the responsibility for the application itself to the cloud service provider. Common examples include software packages such as Microsoft Office 365 or Google Docs.

Figure 1.1 (source: the Dr. Derrek's guide[5]) illustrates how common tasks are distributed differently between cloud service providers and users in the different service models.

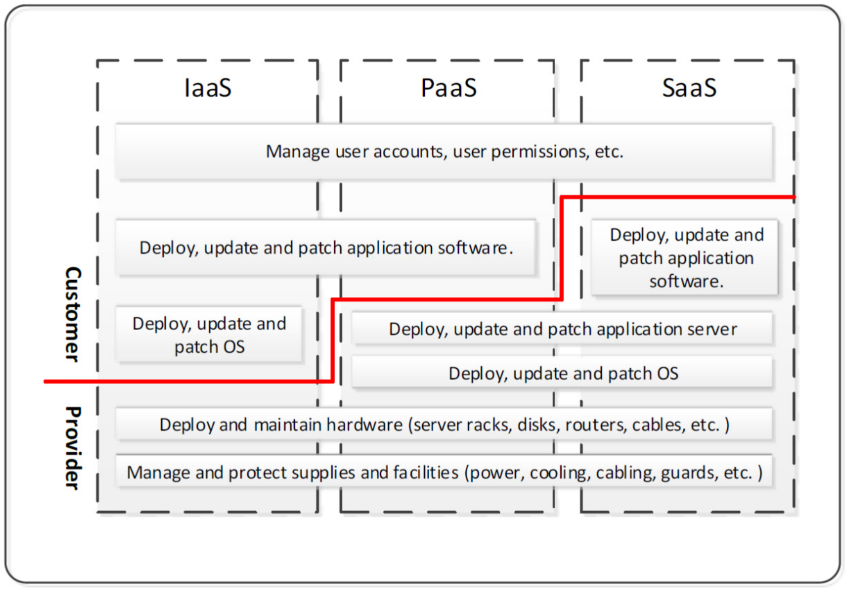
IaaS
PaaS
SaaS
Manage user accounts, user permissions, etc.
Deploy, update and patch application software.
Deploy, update and patch application software.
Deploy, update and patch OS
Deploy, update and patch application server
Deploy, update and patch OS
Deploy and maintain hardware (server racks, disks, routers, cables, etc.)
Manage and protect supplies and facilities (power, cooling, cabling, guards, etc.)
Customer
Provider

Figure 1.1: Customer and Provider Responsibilities in Different Service Models

Note how certain responsibilities always lie with the cloud service provider (specifically, the underlying facilities and hardware), while user management typically lies with the cloud service users. The responsibility for everything in between - operating systems, platform components like application servers, and the actual application software - depends on the cloud service model. Chapter 4 discusses cloud service models in more detail.

1.1.2 Cloud Deployment Models

The discussion so far implicitly assumed the public cloud deployment model, in which a (typically large) organization such as Amazon, Google or Mi-

[5]Dr. M.A.C. Dekker, Dimitra Liveri: "Cloud Security Guide for SMEs", European Union Agency for Network and Information Security

crosoft acts as the cloud service provider for a large number (typically many thousands) of clients organizations. This is probably the most common cloud deployment model, since all of the benefits discussed in the next section apply. In the rest of this section we present the other cloud deployment models illustrated in Figure 1.2 (source: the Cloudiofy website[6]).

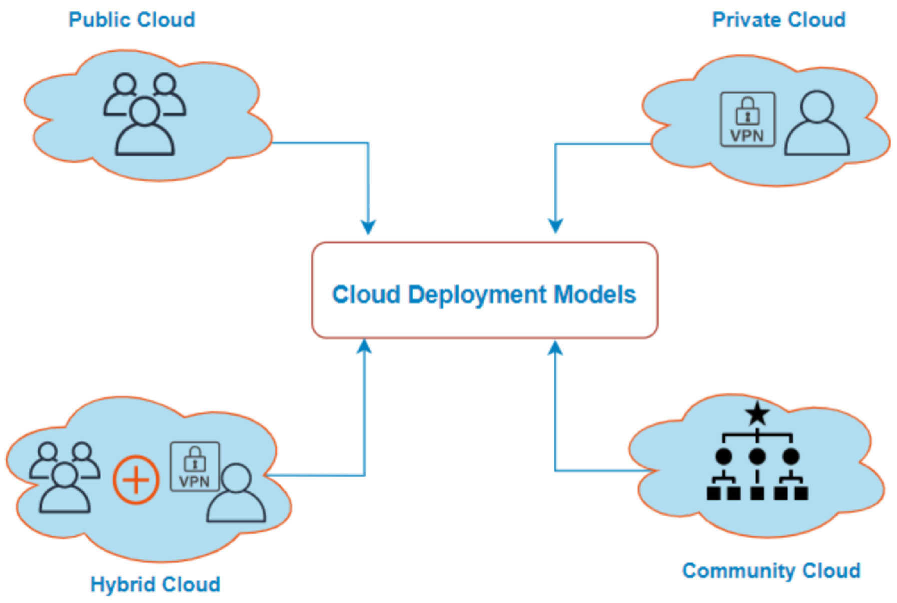

Figure 1.2: The Cloud Deployment Models

Public cloud computing inherently has multitenancy - which means that customers of different organizations share the same physical hardware. For example, data of multiple customers might be stored on the same physical hard drive, and virtual machines of multiple customers can be hosted on the same physical server.

Theoretically, multitenancy should not be a problem - if everything works as intended, the virtual resources of different customers are strictly isolated from each other. In practice, this is mostly the case. However, bugs in the implementation of operating systems, virtualization software or even hardware (see e.g. the Meltdown and Spectre website [7]) can break this isolation. Therefore, public cloud use always comes with a small risk that the confidentiality of data might be violated. This is one of several reasons why

[6]The Cloudiofy project: `https://cloudiofy.com/types-of-cloud-computing/`
[7]Meltdown and Spectre: `https://meltdownattack.com/`

public cloud use might be unwise (or prohibited by government regulations) for organizations that handle particularly sensitive kinds of data.

If this is the case, there are two main alternatives to public cloud use: Semi-private or private cloud. **Semi-private cloud** is actually quite similar to public cloud, in that a service provider typically hosts the virtual resources of many different customers. The key difference is that there is no multitenancy. In other words, physical resources are strictly separated between customers. The obvious advantage to public cloud computing is that it is more secure - the isolation between different customers is stronger. The main disadvantage is that costs are higher, since the cloud service provider cannot maximize the utilization of physical resources as well as it can in public cloud use. For example, if the cloud provider uses servers with 64 CPU cores, then four different customers who all want to use virtual machines with 16 CPU cores each can share the same physical server in a public cloud setting. In a semi-private cloud setting, the provider has to have (at least) one server per customer, which in this example means four servers, and consequently four times the cost.

Another problem with semi-private cloud is that it still carries security risks. In particular, the customer has to trust the cloud service provider, who hosts (and consequently has access to) all the virtual resources of the customer. Either negligence/incompetence or malevolence of the (public or semi-private) cloud service provider could put customer data at risk. This is one of the reasons why some organizations choose (or are forced by regulations) to host their own private cloud.

In a **private cloud**, the cloud service user and cloud service provider are the same organization. The NIST definition of cloud computing from above still applies - "a shared pool of configurable computing resources [...] that can be rapidly provisioned and released with minimal management effort". So rather than having separate physical servers for different IT systems, the organization has a number of physical servers on which all IT systems run in a virtualized way, e.g. as virtual machines. Within the limitations of the physical machines, this has many of the benefits of public cloud computing as discussed in the next section. For example, the virtual hardware of individual IT systems can rapidly be scaled up and down based on demand (as long as the physical servers have capacity left).

The main advantage of private cloud computing is that the organization has full control over its IT resources. It does not have to rely on a service provider to keep its data secure. This is closely connected to one of the main disadvantages - the organization has to create, run and maintain its own cloud infrastructure, which requires a significant amount of expertise and effort. Another key disadvantage compared to public cloud computing is that the scalability is strictly limited by the physical hardware that the organization has (while large public cloud providers typically offer virtually limitless resources - in other words, it is safe to assume that an individual customer will run out of money to pay for resources before the provider runs out of resources to offer).

It should be noted that an organization choosing a private cloud has different options of doing so, with different levels of required in-house expertise. The private cloud platform OpenStack distinguishes three approaches of running a private cloud[8]:

- a managed cloud, where a service provider helps to create and manage the private cloud infrastructure;
- the use of a supported private cloud distribution which takes care of tasks such as testing, bug fixing and packaging of the cloud infrastructure software;
- a *Do It Yourself* (DIY) approach in which all these tasks fall to the organization itself.

The **hybrid cloud** model combines some of the advantages (and disadvantages) of private and public cloud computing. In the hybrid model, IT resources are normally hosted in a private cloud. However, if a peak in resource use occurs, the system is set up to fulfill the demand that exceeds the locally available resources by accessing a public cloud provider. This is sometimes also called cloud bursting. It might seem as if the hybrid model mostly combines the disadvantages of both the private and public model - the organization has to maintain its own cloud infrastructure, and if cloud bursting is required, the data is no longer completely under the control of the organization. However, it should be noted that steps can be taken to mitigate the resulting risks. For example, cloud bursting could be limited to applications with less critical data. Some of the main advantages of using a hybrid cloud model are that an organization is not completely dependent on a provider, that it can maintain significant IT expertise in house, and that often already available physical IT resources can be continued to be utilized without running the risk of running out of capacity.

There is one more cloud deployment model - **community cloud**. In a community cloud, a number of organizations share cloud infrastructure

[8]"OpenStack: The Path to Cloud": `https://www.openstack.org/assets/path-to-cloud/OpenStack-6x9Booklet-online.pdf`

that is privately hosted by one or several of them. This model is often used by similar organizations that do not compete strongly against each other. For example, a number of school boards or a number of colleges and universities might decide that privacy regulations in their jurisdiction make public cloud use impractical for them. On the other hand, each individual organization might not have (or might not want to spend) the resources to host their own private cloud. In such a situation, it makes sense for them to run a shared community cloud, which allows them to run the cloud in a way that follows all the regulations in their industry and jurisdiction. An example for a community cloud in the Canadian province of British Columbia is EduCloud[9].

Deciding which cloud deployment model is best for an organization is no trivial task. Factors to consider include:

- **Regulatory and compliance requirements:** Privacy and security requirements by government regulations might allow certain models such as public cloud computing only under very specific conditions.

- **Financial considerations:** The required initial investment amount and the ongoing costs differ significantly between the different deployment models.

- **Elasticity of resource demand:** High fluctuations in IT resource needs tend to favour public cloud use, and make private cloud use less efficient.

- **Geographical access patterns:** If accesses are only from one geographic location, then a private cloud data center in that location might offer the lowest latencies. However, if accesses are from all over the world, using the worldwide network of server centers operated by large public cloud service providers might be preferable.

1.2 Benefits of cloud computing

Two of the main benefits usually associated with cloud computing are elasticity (i.e. the ability to scale resource use up and down rapidly) and cost savings. However, there are additional benefits that might be even more important in certain circumstances, including increased reliability, increased

[9] `https://www.bc.net/service-catalogue/educloud-server`

performance, and a significant reduction of in-house IT needs. The following subsections discuss these benefits (which are all from the perspective of the cloud user) in some more detail. The end of this section then discusses the main benefits for cloud service providers.

1.2.1 Elasticity

The concept of elasticity can easily be visualized with a rubber band that can be stretched to get longer, but that also returns to its previous length rapidly if it is no longer stretched out. Cloud computing offers great elasticity of resource use - a cloud user can rapidly increase or decrease resource use, and only has to pay for the actual resources that are used over time[10]. Elasticity in public cloud computing is even better than in the rubber band analogy - while rubber bands usually break when they are stretched too far, the resources that public cloud service providers offer are practically unlimited: While a rubber band probably breaks when it is stretched to say ten times its usual length, a cloud user can easily scale from a single virtual machine to hundreds or even thousands of virtual machines within minutes, as long as they can afford the cost.

Elasticity is very useful in many different scenarios. For example, online stores have significant peaks and troughs in demand throughout the day (e.g. most people shop in the evening, some during the day, but very few at night) and throughout the year (e.g. very busy for a few weeks before Christmas and on certain days like Black Friday or Boxing day, but much less busy at other times of the year). For example, the website of the online retailer BestBuy has an annual peak traffic in the Christmas shopping season that is more than seven times as high as the average traffic[11]. If the IT infrastructure behind the website was run on servers owned by BestBuy that were only used for this purpose, three quarters of the capacity of these servers would be idle for most of the year.

Another example for the usefulness of elasticity is if demand peaks suddenly. This might happen if a previously not very well known website is mentioned on a popular website or even on a TV show. In such a case, the number of accesses can suddenly grow by orders of magnitude, overwhelming any single server.

[10]In contrast to scalability, which is about the ability of a system to handle a higher workload if it is run on more powerful hardware, elasticity refers to the ability to scale resources up and down dynamically in response to the workload.

[11]Joel Crabb, "The BestBuy.com Cloud Architecture", IEEE Software, March/April 2014, pp. 91-96

A final example for the benefits of elasticity might be a system that allows senior managers to run complex analyses of large amounts of data. When such an analysis is run, it is highly desirable to have the results available within a short period of time (minutes or hours). This requires massive computational power. However, since managers will only spend a fraction of the time running such analyses (and only do so during the day on work days), the underlying IT systems will be idle for the vast majority of the time. Once again, the elasticity of cloud computing can solve this problem - when an analysis is run, dozens of powerful virtual machines can be spun up to solve the problem in parallel. Once the analysis is complete, they can be shut down equally rapidly, leading to a quick result at a low cost.

1.2.2 Cost Savings

The benefit of cost savings is closely related to elasticity. Given that a cloud service user only pays for the actually consumed resources, the more resource use fluctuates over time, the more money can be saved by using resources in the cloud. This is because if self-owned IT resources are used instead, they have to be powerful enough for the peaks of resource use, and consequently sit partially idle during all other times.

But how is it that cloud service providers can offer IT resources at rates that are low enough to actually lead to cost savings? There are two main factors at play: Firstly, the peaks of different cloud service users occur at different times. While certain types of systems (e.g. e-commerce systems) will likely have high resource demand at the same time (e.g., on big shopping days like Boxing Day or Black Friday), cloud service providers typically host many other kinds of systems as well. For example, many customers host their corporate IT systems (e.g. accounting systems, HR systems, ERP systems) with public cloud service providers as well. These systems typically have their peak loads at other times - during normal working days, when the demand for e-commerce systems is lower. Another reason for different peaks is that peak usage typically occurs at certain times of the day. For example, streaming video services are most in demand in the evening hours. Since these types of services are hosted and used around the globe, the peak demand occurs at different times in different locations due to time zone differences, once again smoothing out peaks. The end result of these effects is that cloud service providers are able to run their servers at a much higher utilization than any individual organization could, significantly reducing their cost compared to individual organizations running their own IT.

Another important reason why cloud service providers can typically offer IT resources at very competitive rates is simply down to economics of scale. The data centers of the the big cloud service providers are massive - probably containing hundreds of thousands of servers[12]. Not only does this mean that the big providers can procure servers, networking equipment etc. at a discount due to the volume they purchase, but it also means that many relatively fixed costs can be spread across a very large number of paying customers. For example, a data center needs physical security measures such as surveillance cameras, security personnel etc. While a large data center probably spends more on these than a small data center, the costs scale much slower than the number of servers. For example, a very small data center with a hundred servers can be managed by a single security guard at a time. A large server center with 100,000 servers probably needs several security guards (to account for the larger size of the center), but likely significantly fewer than a thousand security guards at a time. Hence, the cost per server for physical security gets lower the larger the server center gets, giving the massive service centers of public cloud service providers a significant cost advantage.

1.2.3 Increased Reliability

Using a public cloud service provider can significantly increase reliability. Public cloud service providers host server centers in multiple locations globally[13]. Hosting an application redundantly in multiple server center locations makes it resilient in the face of natural disasters such as major storms, floods or earthquakes.

The elasticity of cloud computing can also increase reliability significantly, as it can be exploited to thwart *Denial-of-Service* (DOS) attacks, in which an attacker floods an application with fake requests in an attempt to make it unresponsive for real requests by actual customers. Given the ability and willingness to spend the necessary amounts, DOS attacks can be made ineffective by scaling the system to as many servers as are necessary to respond to all requests. Additionally, public cloud providers have network-based mechanisms to detect and mitigate DOS attacks.

[12]Details are generally not published by the big cloud service providers, as they represent proprietary and competitive information. The following video "Inside a google Data Center" gives a good idea of the scale of a modern cloud data center (`https://www.youtube.com/watch?v=XZmGGAbHqa0`).

[13]For a map of locations of Google Cloud, see `https://cloud.google.com/about/locations#regions`

A final argument for increased reliability is that the expertise in fields such as IT security is likely higher in a public cloud service provider than in most other organizations, especially if they are from industries other than IT.

Some of these benefits (such as increased reliability at reduced costs) are particularly strong for small companies. Imagine a small startup company that offers a specific service online. Because the customer base is initially very small, a single server might be capable of hosting the service. Therefore, the service could be hosted very inexpensively by dedicating a single machine (possibly just an inexpensive PC) in the office of the startup to running it.

One massive disadvantage to this approach would be that reliability of such a service would be poor. A number of problems could lead to the service being unavailable for customers: For example, a technical failure in the single server, a network component (e.g. router or switch) failure, a power outage, an outage of the internet service provider, a disaster such as a fire in the office building etc. The obvious answer to all these risks is redundancy. So instead of a single server, at least two servers should be used, so if one fails, the other one can continue to respond to customer requests. Similarly, all networking equipment should be redundant as well. For extra resilience, the two servers should actually be in two separate locations, so that a single disaster (such as a fire or localized flooding) cannot bring the whole service down. On top of this, it would be preferable to have redundant power supply by two independent providers, if possible, or at least uninterruptible power supply infrastructure (battery packs and diesel generators) as well as redundant internet connections.

It should be clear that such a setup would be very expensive to implement for a small startup company. However, there is an alternative: renting two small virtual machines in two separate server centers of a public cloud service provider gives the same level of reliability as the expensive setup described above, at a very small fraction of the cost.

1.2.4 Increased Performance

Using cloud computing can lead to a performance increase in the hosted service. This is mainly due to two reasons that have already been discussed in this chapter: Elasticity and the fact that the big public cloud service providers operate multiple data centers across the globe.

If a cloud application is designed to automatically scale up the number of servers it runs on as requests increase, consistently high performance (e.g. measured in short average response times) can be ensured much more easily than with a system that is hosted by an individual organization. The global distribution of data centers by the big cloud providers is not only for redundancy and regulatory reasons, but also to reduce the network latency for end users. For example, if a globally operating company responds to requests from around the globe with a website that is only hosted in their own data center in North America, then performance in e.g. Australia is significantly worse because of inevitable network latencies. However, if hosted e.g. in Google Cloud, then requests in that region can be routed to Google's data center in Sydney, drastically reducing latencies.

It should be noted that increased performance is only achieved if the cloud systems are well designed. Incidentally, this is true for all the other benefits as well - a badly designed cloud application that does not use redundancy well is consequently not particularly reliable, one that does not scale automatically does not take advantage of elasticity etc.

1.2.5 Reduced In-House IT Needs

A final major benefit of using public cloud computing rather than hosting all IT in-house is that it reduces how many in-house IT personnel are needed. The magnitude of this reduction depends on a number of factors, including the service model that is used: If the predominant model is Software as a Service, then very limited in-house IT staff is needed. On the other hand, if Infrastructure as a Service is the main approach, a wide range of in-house IT specialists from various kinds of system administrators to software developers are still needed, and only roles directly involved with the underlying hardware are eliminated.

Reducing in-house IT staff can be advantageous in several different ways. It often comes with cost savings, because (e.g. due to economics of scale) a public cloud service provider can typically perform the same work more cheaply. Additionally, it allows companies, especially those that are not in the IT industry themselves, to focus on their actual business. A related point would be that such companies often have difficulty attracting and retaining highly qualified IT personnel.

This concludes the discussion of some of the main benefits of cloud computing from the perspective of a cloud user. The arguments above implicitly assumed the use of public cloud computing. However, to a lesser degree, many of the points above still apply in other cloud deployment models such as private cloud. For example, while elasticity in the private cloud model is strictly limited by the number of physical machines the organization has, using a private cloud platform still allows the different applications of the organization to scale up and down flexibly, as long as their total resource needs do not exceed what is available. The main exception to the rule that the benefits of public cloud computing apply similarly (if to a lesser degree) to private cloud computing as well is that running a private cloud can be highly complex, which implies that it still requires very significant in-house IT expertise.

1.2.6 Provider Benefits

The most obvious benefit of a cloud service provider is that offering cloud services can be highly profitable: For example, in the second quarter of 2021, Amazon Web Services (AWS) posted revenues of almost \$15 billion (up approximately 27% from the previous year) with profits (operating income) of over \$4 billion[14].

These numbers are remarkable in a number of ways: Not only is the size of the cloud computing business significant, but also its growth rate and its profit margins. And the latter is true even though there is strong competition between the big cloud service providers (Amazon AWS, Microsoft Azure and Google Cloud). These numbers in their own right can serve as an indication that cloud computing is beneficial - why would customer organizations spend so much on services that they did not find useful?

Besides the profitability of the business, there are other benefits for cloud service providers. For example, most cloud service providers need significant IT resources for their own IT needs. Offering spare capacity to cloud service users allows them to increase the utilization of IT resources they already have. Additionally, offering cloud services can be a very effective strategy to retain customers. This observation leads directly to the drawbacks of cloud computing.

1.3 Drawbacks and Challenges of Cloud Computing

Even though the large economic success of cloud computing hinted at in the last chapter is a strong indication that the advantages of cloud computing often outweigh its disadvantages, there clearly are drawbacks of challenges related to the use of cloud computing. This section discusses some of the most important ones, including security and privacy concerns,

[14]Amazon Q1-2020 Earnings Release: `https://s2.q4cdn.com/299287126/files/doc_financials/2021/q2/AMZN-Q2-2021-Earnings-Release.pdf`

vendor lock-in risks, cost and cost management issues, lack of expertise, performance concerns and other organizational issues.

1.3.1 Security and Privacy

Security and privacy are probably the most commonly quoted concerns around cloud computing. They are really two distinct issues, but strongly related, so we discuss both together here. More details are discussed in Chapter 6 about security.

Security can be defined as the combination of confidentiality, integrity and availability. For example, for a database to be secure, access to its contents must be limited to people authorized to access it (confidentiality), nobody unauthorized must be able to delete or change data in the database (integrity) and the database must actually be available to authorized users (availability). Availability might sound like a non-security issue, but without it, the other two are trivial to achieve (lock a hard drive with the data in a big safe, and confidentiality and integrity are achieved). Similarly, in practical terms, an organization can get into trouble if either of the three components is violated - if customer data is disclosed, if it is manipulated, or if it cannot be accessed.

Privacy has an even wider range than security, and how much it is valued strongly depends on the laws and cultures of different countries. For example, while a disclosure of a complete list of movies watched by a person on a streaming service might not cause any direct (e.g. financial) harm, most people would consider it a violation of their privacy if such information was publicly accessible.

With regard to both security and privacy, a cloud service user has to trust the cloud service provider. Since (irrespective of the cloud service model) the cloud service provider operates the physical hardware on which the applications and/or data of the cloud service user ultimately reside, they have physical access to these applications and data. Consequently, a cloud service user has to trust the competence and benevolence of the cloud service provider, meaning that it does not accidentally or intentionally expose confidential data to unauthorized parties etc.

A related point is that countries often have regulations limiting cloud use for certain industries that handle particularly sensitive data such as in the health care sector. For example, colleges and universities in the Canadian province of British Columbia are generally not allowed to store data with cloud service providers that might be hosted outside of Canada, since data protection and privacy laws e.g. in the USA are less stringent than in

Canada. Compliance with such regulatory requirements can be challenging for organizations.

Security and privacy questions should carefully be analyzed by any organization that wants to use cloud computing. There are many measures that can be taken to mitigate risks. In particular, the use of strong encryption can reduce the risk that sensitive information gets disclosed due to e.g. a provider error. Additionally, it should be pointed out that the use of cloud computing can also increase security if it is done properly - for example, high availability can more easily be achieved by using a cloud provider with multiple secure data centers. Finally, it is important to recognize that cloud computing is not appropriate in all situations. For example, the control software of a nuclear power plant should not be run in any cloud system that can be accessed from the outside, since its security is of such critical importance.

1.3.2 Vendor Lock-in

Vendor lock-in describes the situation that an organization which hosts IT resources at a cloud service provider tends to be locked into that provider, meaning that the level of difficulty of switching to a different provider would fall somewhere into the range from hard and expensive to practically impossible. This might not be a problem as long as the organization is happy with its cloud service provider. However, this dependency on a specific provider can become very problematic if there are significant disputes about e.g. provided services or billing, if the provider goes out of business, or simply if better and cheaper other providers become available.

Vendor lock-in is nothing new in the IT world. For example, most organizations store their data in commercial relational databases. These databases typically follow the SQL standard, and should therefore be easily interchangeable. However, practically all commercial database vendors have their own proprietary extensions of SQL. The justification for these extensions is typically that they add useful functionality that has not (yet) been standardized in SQL. However, customers who utilize these extensions will find it significantly harder to switch to a different database vendor.

In cloud computing, the degree of vendor lock-in experienced depends on several factors, including the service model chosen. Because there is very little cross-vendor standardization in cloud computing overall, a (slightly oversimplified) way to judge the severity of vendor lock-in could be to look at how many responsibilities lie with the cloud service provider vs. user.

For example, if “Infrastructure as a Service” is used, the responsibility for operating systems, platform, applications etc. lies with the cloud service user. Therefore, switching to a different cloud service provider would probably be easier (as long as it offers similar virtual machines etc.) than for an organization that uses “Platform as a Service”, which might have to change its applications significantly because no other cloud provider offers exactly the same platform.

There are different strategies to deal with vendor lock-in, and they all come with difficult tradeoffs between flexibility, cost etc. For example, an organization could decide to host its IT resources with multiple public cloud providers. While this creates significant extra effort and cost, it makes it much easier to shift resources away from a provider if necessary. Another approach (probably more suitable for smaller organizations, who cannot or do not want to afford using multiple cloud providers) would be to wisely choose the cloud features used, and refraining from using features that are not offered in a similar form by other providers.

1.3.3 Cost and Cost Management

The attentive reader might be surprised to see cost as one of the disadvantages or challenges of cloud computing, after it was listed as one of the main advantages earlier. Just like with many other aspects of cloud computing, the answer here is “it depends”.

While it is generally true that cloud service providers offer IT resources at cheaper rates than their typically much smaller customer organizations can achieve when running their own IT, this really depends on several assumptions. In particular, if an organization already owns a lot of IT resources (servers, networking equipment, but also server software licenses) when it decides to move to the cloud, then the ongoing costs of cloud computing might exceed the cost of operating the already owned equipment. This is especially true if the IT needs of the organization are very constant, allowing high resource utilization and implying no need for spare capacity that sits idle most of the year.

Another issue is not the absolute level of cost, but the fact that costs are much more volatile in cloud computing. An organization that owns and operates its own IT has very predictable and fairly stable IT costs - besides some extra electricity consumption, servers (and the personnel operating them) cost roughly the same whether they are running near peak capacity or idly. In contrast, costs in cloud computing are highly volatile. Since users only pay for what they use, if an application has four times the load in the busy Christmas shopping season than it has in say February, then the costs will also differ by a factor of four. Even if the overall costs for the whole year are typically lower if cloud computing is used, this volatility can be problematic for organizations that are not used to it.

A final point is that costs can spike unexpectedly, either due to misconfigurations that scale systems up unnecessarily, or because of applications that correctly respond to an unexpected volume of requests (e.g. due to a denial of service attack) by scaling up massively. This is also a potential issue for learners who want to try out a cloud computing platform by using the usually available trial accounts that provide a number of free credits, but require a credit card that is automatically charged if those credits are exceeded: It occasionally happens that students forget to turn off e.g. a group of virtual machines, in which case the costs can build up to hundreds of dollars in just a few weeks. Thankfully, some cloud providers have apparently recognized this issue and are now offering special accounts for educational purposes that can be used without entering a credit card, and which simply deactivate services once the provided funding runs out.

In summary, while cloud computing is often cheaper than operating ones own IT infrastructure, all the costs (initially and ongoing) should be analyzed carefully upfront. If cloud computing is used, its ongoing costs should be monitored closely to avoid surprises. Management also has to be on board and move away from traditional thinking in fixed budgets. If for example a well configured online store incurs increasing hosting costs by the cloud service provider, this is not a bad thing, but actually desirable, because the reason should be that the online store is busier (and consequently hopefully generates more revenues and profits).

1.3.4 Lack of Expertise

Lack of cloud expertise is another often quoted issue that organizations considering cloud computing face. This is not particularly surprising, given that cloud computing is still relatively new, and growing at a rapid rate. Only quite recently did the first universities and colleges start offering degrees or specializations in cloud computing. The fact that cloud expertise is in short supply can be illustrated by the fact that cloud computing was the most sought after hard skill on the platform LinkedIn in 2019[15], and by an analysis showing that cloud computing professionals earn the highest

[15]`http://shorturl.at/cBNPY`

average salaries in the IT function aside from IT executives[16].

Not every organization using cloud computing needs the same level of cloud computing expertise. Both the number of employees with cloud computing skills, as well as the specific knowledge required depend strongly on the cloud deployment and service models used. For example, an organization that chooses a hybrid cloud computing approach to run different applications with a range of different cloud service models obviously needs significantly more employees with high levels of cloud expertise than an organization that primarily uses *Software as a Service* in a public cloud setting.

Still, irrespective of the models chosen, to get the full benefit from using cloud computing, it is highly desirable that employees in many different job functions understand cloud computing, from specific technical expertise in technical operators to strategic understanding of the business opportunities afforded to an organization by the flexibility of cloud computing in executives.

There are several ways how organizations that do not have the cloud computing expertise they require can mitigate this situation, including training their current employees, hiring additional employees and temporarily using outside consultants. Additionally, deployment and service models that make sense with the given level of existing expertise should be chosen.

1.3.5 Performance Concerns

Performance concerns specifically about public cloud computing come primarily from two main sources: Network limitations and performance implications of shared physical resources.

With regard to network limitations, the main issue is generally network latency. The further away a server responding to client request is located, the longer the time signals have to travel, and, usually even more importantly, the more "network hops" have to be performed, i.e. the more intermediate switches, routers etc. that each add a little bit of delay are involved. It highly depends on the situation whether these latencies are problematic. For some applications (e.g. transfer of large files), delays of even a few seconds hardly matter to users, while for other uses (e.g. video conferences) even delays of just a few tenths of a second are highly inconvenient. Network bandwidth is usually less important, unless massive amounts of data have to be transferred. If that is the case, and if the Internet connections of

[16] `https://www.crn.com/news/global-it-salaries-hit-new-high-2019-it-skills-and-salary-report`

some of the involved parties are not high speed, cloud computing might in fact be not suitable.

The second performance concern comes from the fact that physical resources such as servers are usually shared by multiple virtual resources such as virtual machines of the same or different organizations. Consequently, it is at least theoretically[17] possible that the performance of an application is degraded by other applications (possibly belonging to other organizations) that run on the same server. Another concern is that most cloud providers do not guarantee the immediate availability of resources in the requested location. In times of generally high resource utilization it is possible that requested resources take a significant amount of time to become available, or are only available immediately in different geographic locations, which could impact the performance of an application that needs to scale up due to high load.

Two ways to address these possible performance issues is to measure performance continuously, and to automatically scale systems up as needed if performance dips. It should be noted that, just as with many other points discussed here, cloud computing can both be a source of and a solution to performance issues. For example, it is quite possible that the average network latency experienced by end users is lower if a public cloud service provider is used, since the provider operates multiple data centers across the globe. Consequently, the average user might be "closer" to a data center of the chosen public cloud provider than to the in-house IT resources of the organization.

1.3.6 Organizational Issues

The final challenge in adopting cloud computing discussed here can be summarized as organizational issues. Often times, companies starting to use cloud computing find that their organizational culture is causing issues. It is beyond the scope of this introduction to discuss all the possible obstacles and their solutions. Instead, the following just gives an example.

A very common obstacle would be resistance to change, which often affects other types of IT projects as well. Such resistance to change can have many different causes, from psychological factors to rational consid-

[17]Technical details of this point are beyond the scope of this introductory chapter. However, it should be noted that modern virtualization and cloud computing platforms generally offer effective means to isolate different virtual resources that share the same physical resources in such a way that the load on one does not affect the guaranteed minimum performance of others.

erations. For instance, an IT manager might resist moving internal IT resources to a public cloud provider because they are not sure that they (and their employees) possess the necessary skills. An additional concern might be that the IT department would likely shrink, since a lot of work maintaining physical IT resources in house would be eliminated. This could result in undesirable layoffs, and also reduce the size of the IT department, which could make the manager afraid of losing status and power in the organization. Finally, the manager could be afraid that their performance evaluation (e.g. as a function of how smoothly the IT operates) could increasingly depend on factors outside of their control (and instead under the control of the cloud service provider).

Given that organizational obstacles to cloud computing are different in each individual case, no single approach for mitigation can be described here. However, achieving buy-in into cloud computing plans by all affected stakeholders is a critical goal, and a lot of honest (two-way) communication is likely an important means to achieve it. Also, getting external know-how will likely help, be in the form of training, new hires, or the use of consultants.

1.4 History

In order to understand the appeal of cloud computing for organizations, it is beneficial to briefly review the history of computing paradigms. This evolution of IT from mainframe systems to modern cloud computing is illustrated in Figure 1.3 (source: the cloud experts website[18]). The remainder of this section discusses the main steps of this evolution.

1.4.1 The Era of Mainframes

We begin with the earliest Turing-complete computers, i.e. computers that could not just execute predetermined computations, but execute general purpose programs. Such computers first became commercially available in the 1950s, and became more widely used in the 1960s. These computers initially did not use integrated circuits yet, but had to be built from combining individual transistors. Consequently, they were very large (think – room sized) and very expensive.

[18]The cloud expert website, `https://www.cloudexpertsweb.com/view_blog.php?blog_id=51`

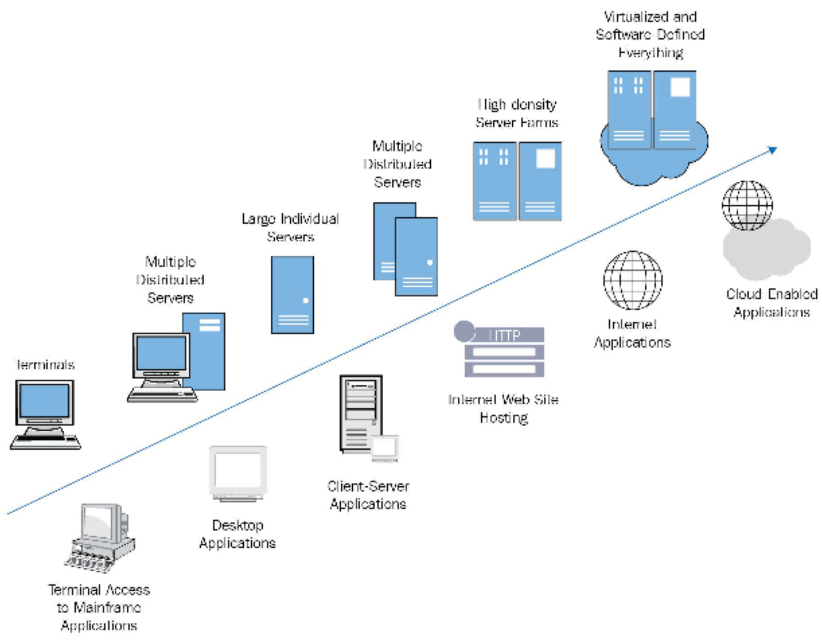
Virtualized and Software Defined Everything
High density Server Farms
Multiple Distributed Servers
Large Individual Servers
Multiple Distributed Servers
Terminals
HTTP
Internet Web Site Hosting
Internet Applications
Cloud Enabled Applications
Client-Server Applications
Desktop Applications
Terminal Access to Mainframe Applications

Figure 1.3: The Evolution of cloud computing

Initially, these computers had to be programmed by manually inputting whatever program one wanted to run, often using sets of buttons or switches to directly manipulate the contents of memory cells. Needless to say, this was a lengthy and error prone process. As a consequence, typically much more time was spent entering (and fixing) the program than actually executing it, resulting in very low effective utilization of these very expensive machines.

Not surprisingly, better approaches to perform I/O were invented in the 1950s, and the first precursors of operating systems appeared. These simple batch systems typically had one (or multiple) punch card readers through which programs could be loaded semi-automatically, so that at the completion of a program the next program was ready to run. This improved the effective utilization of the computer significantly. The usual way to use these machines was to manually punch the correct holes for the desired program into multiple sheets of paper, drop them off at the operator of the computer, and to pick the results up at a later time (often hours or days later, depending on how many other programs were ahead in the batch queue) once the program had been executed.

These simple batch systems significantly reduced the time wasted between programs by eliminating lengthy setup time of programs, and removing the need to schedule specific time windows for access to the computer in advance. However, CPU utilization remained low, because the CPU had to sit idle during I/O operations. Consider a typical batch program that might be run in a large organization such as processing hourly payroll. In such a program, the program logic might consist of a single loop over all hourly employees. For each employee, the relevant information such as hours worked, hourly rate, factors influencing payroll deductions etc. would be read in from some input device (a card reader, a tape, or even a disk drive). Once the employee data was loaded, all the necessary calculations were performed in memory, and then the result (e.g. a printed check) was output. Now assume that the CPU runs at 10 MHz, the calculations for each employee on average take 1,000 instructions, and the read and write operations for each employee take a total of 10ms. Under these assumptions, the time spent on CPU processing for each employee is 0.1ms, and consequently the CPU sits idle waiting for I/O to complete about 99% of the time.

To alleviate this issue, and to significantly increase CPU utilization, multi-programmed batch systems became common in the 1960s. In these systems, the computer keeps multiple batch programs in memory. Once a program needs to do I/O, its execution is paused, and another program is executed. Once the I/O operation is complete, the program becomes ready to be executed again, and is scheduled to be resume execution once the CPU becomes free (e.g. once the other program needs to perform I/O itself). Such multi-programmed batch systems were used for a long time (and in fact, are still in much more widespread use than one might suspect).

The next major innovation in computing was the invention of time sharing systems, which began to be widely adopted in the late 1960s . Just like multi-programmed batch systems, time sharing systems keep multiple programs in memory. However, rather than typically running an individual program until it cannot continue (e.g. due to an I/O operation), with a focus on maximizing CPU utilization and system throughput, these time sharing systems performed each program only for a short time period (called time slice) at a time. For example, assume that each program is executed for 100 milliseconds at a time, and that there are ten active programs. In such a scenario, each program gets to execute for 1/10 of a second every second. Even at a CPU speed of just say 10MHz, this means that each program can execute one million instructions per second. Consequently, for each of the ten programs, the computer effectively performs as if it had a CPU on its own that has the performance of a 1MHz CPU.

A computer running such a time sharing system was usually shared by many users in a company, or a large department in a company, and accessed from so-called "dumb terminals", consisting of not much more than a (single color, text only) screen and a keyboard that were connected directly to the computer in the same building.

1.4.2 Personal Computers and Client-Server Systems

Significant technological progress made more and more complex integrated circuits possible. In the late 1970s, this led to the emergence of Personal Computers (PCs). These machines were typically significantly less powerful than the mainframes used in time-sharing systems. However, their price was also so much lower that it became affordable to purchase a computer for each employee, or even for hobby use at home. When IBM introduced the IBM PC in 1981, the adoption of personal computers grew massively. Initially, these personal computers were often not connected through a computer network. Instead, they ran standalone applications such as spreadsheets or word processing. Data was typically exchanged using physical media such as floppy disks.

Of course, for many applications, data needs to be exchanged between multiple different users. A typical example might be a database system that stores inventory information of a manufacturing company. Such a system typically needs to be accessed by multiple employees in different departments such as the warehouse, purchasing, sales, product assembly and accounting. Such applications can be implemented following the client-server paradigm: One (or multiple) centralized servers, running on more powerful computers, are accessed by a number of client PCs. Application-specific software runs on both servers and clients which are connected through a computer network, such as Token Ring or Ethernet (the latter of course becoming the de facto standard for wired computer networks in the 1990s).

There are two main architectures for client-server systems: 2-tier (see Figure 1.4) and 3-tier (see Figure 1.5) architectures (source for both figures: the Guru99 repository [19]). The difference lies in how the different parts of the application - user interface, business logic and storage, are distributed between client and server. While at least part of the user interface (process-

[19] The Guru99 repository, `https://www.guru99.com/dbms-architecture.html`

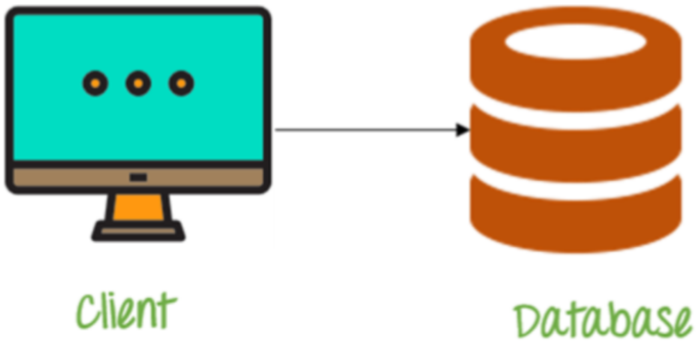

Figure 1.4: The 2-tier architecture

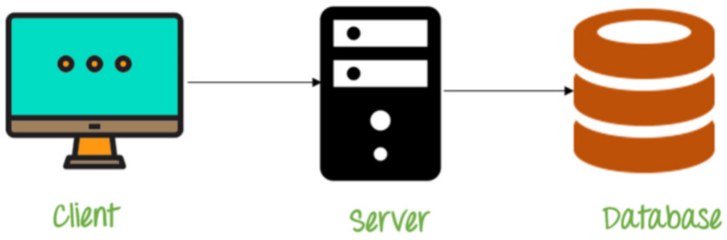

Figure 1.5: The 3-tier architecture

ing user inputs, displaying information) always has to be at the client machine, and at least part of the storage function (actually storing the data that is shared between all the clients involved) has to be located at the server, the business logic can be on one or the other, or distributed to different degrees between both. An example for business logic might be the calculation of prices, which often follows complicated rules for discounts, applying taxes, shipping rates etc.

A disadvantage of 2-tier systems is that having business logic on the client side makes the client component rather heavy (often called fat clients), which might require powerful client PCs, and can easily cause problems if different users running different versions of the client software try to work together. On the other hand, keeping most of the business logic on the server side tends to overload the server, which of course also has the responsibility to store the data, and therefore can become a performance bottleneck for the whole application. 3-tier architectures address these issues by introducing an additional server-side tier, so that the client only has to deal with the user interface, one server tier is only responsible for the business logic, and the second server tier only takes care of storage.

No matter which architecture is chosen, various problems remain. Client computers typically use custom software that has to be updated regularly to fix bugs and to implement new features, creating potential issues if such updates are not performed on all the client computers of an application at the same time. The server side faces additional issues. In particular, in order to ensure that the failure of a single machine (a server) doesn't make the whole application unavailable for all its users, all the components on the server side have to be implemented redundantly. Additionally, many systems do not have a constant workload, but one that fluctuates over time. In order to work without interruptions, the server side has to be powerful enough for the highest required performance level, and is consequently partially idle for the majority of the time, wasting additional resources (in addition to the often idle redundant systems). And this is true for each of the applications used in an organization. Since organizations typically have multiple different applications that are used by different corporate functions (such as sales, purchasing, finance, accounting, research and design etc.), and that are created at different points in time, using whatever approaches and technologies are considered state-of-the-art at those times, this often results in highly complex, hard to maintain and under-utilized IT landscapes in organizations. For example, it is not uncommon for a company to have a few applications running on legacy UltraSparc servers running Solaris Unix, some Windows Server based applications (typically using different, often obsolete versions), in addition to several other applications running on various Linux platforms. To ensure sufficient scalability and fault tolerance, several physical servers for each application have to be maintained.

1.4.3 Grid Computing

At the end of the 1990s, we have seen a considerable increase in commodity computer and network performance, mainly as a result of faster hardware and more sophisticated software. Nevertheless, there were still problems, in the fields of science, engineering and business, which could not be solved effectively with the generation of supercomputers used at that time. In fact, due to their size and complexity, these problems were often numerically and/or data intensive and required a variety of heterogeneous resources that were not available from a single machine. A number of teams had conducted experimental studies on the cooperative use of geographically distributed resources conceived as a single powerful computer. This new approach was known as *Grid Computing*.

The main problem about running tasks on this platform is related the heterogeneity of the machines involved in terms both of hardware and software (including the operating system). At that time users/developers who wanted to submit their tasks had to write applications able to face various kind of failures (hardware, software and network) that also had to be compatible with the operating systems and libraries provided by the computers involved in the grid. As a matter of fact, modern virtualization techniques were not available at the time, so for example, developers had to write two version of their application, one for Unix machines and one for Windows machines, if they wanted to exploit machines from both operating systems.

Once virtualization had reached good levels of both performance and stability, the Grid Computing platform was gradually abandoned in favor of cloud computing, where developers can decide in which operating systems their application must be executed.

1.4.4 Virtualized Data Centers

Virtualization technology, which makes it possible to run multiple so-called virtual machines on a single physical computer, became widely used in the 2000s. Each virtual machine has its own (often different) operating system installed, and for each operating system and its applications the virtual machine behaves as if it was a separate physical machine. For example, a physical server with 16 CPU cores and 64 GB of physical memory can run four virtual machines with 4 virtual CPUs and 16 GB of memory each. For the software (operating system and application software) in each virtual machine it appears as if it was running on its own physical machine with 4 cores and 16 GB.

Virtualization has three key aspects - isolation, encapsulation and compatibility. Isolation describes the fact that virtual machines on a single physical server behave as if they were in fact on separate servers - for example, errors in one VM do not affect other VMs. Encapsulation describes that the entire state of a virtual machine (including virtual hardware, installed operating system, installed applications and all their data and state) are simply a (large) file on the machine hosting the VM. This makes it easy to create backups of VMs and to move them between servers. Finally, compatibility means that the operating system and software in a virtual machine is decoupled from the underlying hardware. Whatever virtual hardware the virtualization software supports it can offer on any physical hardware that it supports. This means that it becomes possible to host legacy hardware in virtual machines on modern physical servers.

These advantages make so-called virtualized data centers very attractive. In such a data center, the servers running business applications do not run directly on physical hardware, but in virtual machines that are hosted on physical servers. This has multiple advantages: Due to VM compatibility, different kinds of servers can be hosted on a single, homogeneous server platform, which reduces complexity and maintenance costs. Encapsulation allows virtual machines to be moved easily between different physical machines, and isolation makes it safe to host multiple different applications in different VMs on the same physical server. Together, these properties significantly reduce costs, as they allow for a higher overall utilization of hardware resources and more efficient server management.

1.4.5 Cloud Computing

While cloud computing data centers extensively use virtualization, cloud computing is more than just using virtualization. In non-cloud virtualized data centers, assigning VMs to physical servers and making decisions about moving them (e.g. due to load changes) are not automated, but typically manually done by operators. Cloud computing adds another layer of abstraction on top of the virtualization layer.

In cloud computing systems, users do not need to know about the underlying physical hardware. This is true no matter what cloud deployment model (e.g. private or public) is used, and irrespective of the cloud service model (e.g. IaaS or PaaS) chosen. The fact that there are different service models is a feature of cloud computing - in simple virtualized data centers, Infrastructure as a Service (i.e. using virtual machines instead of dealing with the underlying physical hardware directly) is the standard mode of operation.

Modern cloud computing depends on fast, reliable and affordable computer networks and mature and efficient virtualization. Consequently, it only became feasible in the last approximately 20 years. It has been argued that cloud computing is in fact just "old wine in a new bottle", implying that it is essentially no different than the first time sharing systems on mainframes over 50 years ago. While it is true that there are some superficial similarities between then and now - such as centralized computing (then on a mainframe, now in a large data center) that is accessed from remote devices (then simple computer terminals, now just about any device with a web browser). However, this architecture back then was based on necessity - if you could only afford a single computer, you had to share it. Modern cloud computing is not a necessity, just (typically) an improvement over more traditional, in-house IT. Also, modern cloud computing platforms offer a wide range of very powerful computing resources - from simple virtual machines to powerful AI-driven programming interfaces.

A discussion of the history of cloud computing would be amiss without mentioning the first public cloud provider. On August 24, 2006, Amazon (then "only" a large online retailer) announced the "Amazon Elastic Compute Cloud". How Amazon decided to not just get into, but invent this new business of cloud computing is an interesting story[20]. In simple terms, executives at Amazon realized that they were very efficient at running the large IT infrastructure they needed to support their e-commerce business. They concluded that it might be a profitable business model to offer access to their IT services to other companies (who were probably less efficient in running their own IT). It probably did not hurt that Amazon's business (and consequently their internal IT capacity needs) fluctuate significantly over the course of a year - with e.g. the Christmas season being very busy for a retailer, while other other parts of the year are much less busy. Hence, it made perfect sense to offer the part of their IT resources that sat idle for a lot of the year to outside customers for a profit. The history of AWS is discussed in more detail in Section 3.4.1.

[20]More info here: `https://techcrunch.com/2016/07/02/andy-jassys-brief-history-of-the-genesis-of-aws/`

1.5 Summary

In summary, while cloud computing can be used in a number of different deployment (public, private etc.) and service (SaaS, PaaS, IaaS) models, its main characteristic and common feature is that computing resources (e.g. storage or virtual machines) can be allocated and deallocated very flexibly from a large pool of such resources. This ability gives cloud service users agility and often makes cloud computing the preferred approach to implement applications, especially since additional benefits such as cost savings and increased reliability can often be realized through cloud use as well. However, cloud computing is no magical solution to all IT problems either - it comes with its own sets of risks and challenges, is not suitable in all situations, and only delivers its potential advantages if implemented well. How to do this is one of the main subjects of the remaining chapters of this book.

Chapter 2

Technical Foundations

The purpose of this chapter is not to give readers all the background necessary to fully understand all aspects of cloud computing - this could fill multiple books, as a basic understanding of a range of topics including computer programming, software design and architecture, operating systems, computer networks and several more would be necessary. Instead, it highlights the main technical foundations that made the widespread adoption of cloud computing possible, and look in some more detail at some technologies that are crucial for cloud computing.

2.1 Networking

Arguably, the single most important technical factor enabling modern cloud computing is the availability of high-speed networking. Moving IT resources (e.g. computing power, storage, or whole applications) to the cloud (i.e., a remote location) is clearly only sensible and desirable if sufficiently high bandwidth and low latency networking connections are available.

For example, in the days of 56K modems, which could only transmit 56 KBit (i.e. 7 KBytes) of data per second, synchronizing local storage (even if it might have only amounted to tens or hundreds of MBytes of data) with a location in the cloud, or running applications in the cloud in the case of Software as a Service, and transmitting the whole user interface across the network was clearly not technically feasible. In contrast to this, while data volumes have grown significantly (as have screen resolutions, which are often Full HD or even 4K now), end-user Internet speeds have grown by an even larger factor of around a thousand since these days. This is true both of cable or DSL based Internet access as well as wireless access (be it through wifi networks or 4G and 5G connections).

Similarly, *Local Area Network* (LAN) speeds to connect servers in data centers, which used to be for instance 10 MBit Ethernet can now easily reach or exceed 10 GBit per second. The bandwidth of wide area networking (WAN) connections, which typically use fiber optical cables, has grown in a similar way.

Network latency is primarily determined by the numbers of "hops" (in particular, routers) that a networking package has to pass through from its source to its destination. This number of hops primarily depends on the geographic distance. Consequently, sufficiently low latencies especially for interactive applications can only be achieved by reducing the distance that network connections have to travel. Public cloud service providers are able to provide most of their customers with fairly local network connections because they have data centers in multiple locations globally.

2.2 Virtualization

Besides inexpensive and high-speed networking, modern virtualization technology is arguably the second major enabler for cloud computing. The development and usefulness of server virtualization depends on another factor - the availability of inexpensive servers with multiple CPU cores and large amounts of memory.

The idea behind virtualization is pretty simple: Computing resources (primarily CPU cores, main memory, storage, networking bandwidth) can be virtually partitioned into a number of virtual machines. For example, a server with 64 CPU cores and 128 GByte of main memory could be partitioned to host 16 virtual machines with 4 CPU cores and 8 GBytes of working memory each. The software (including the operating system) running within each of these virtual machines could not distinguish this situation from one where it runs on an actual physical machine with only 4 CPU cores and 8 GBytes of memory.

Virtualization is beneficial in many different contexts and for a number of reasons. For example, a web developer might use a number of virtual machines to test the application under development on different operating systems and browsers. Without virtualization, a separate physical machine per test system would be required, significantly increasing cost and space requirements. Similar benefits apply in data centers, where consolidating a number of different servers, which, especially if they are older, can easily be run as multiple virtual machines on a modern server, can significantly reduce cost factors such as server maintenance needs, space requirements, power consumption etc.

2.2.1 Virtualization Technology

In virtualization, a hypervisor (also called virtual machine manager) is used. The hypervisor is a piece of software that sits between the physical hardware and the virtual machines. To each individual virtual machine it presents a possibly different virtual hardware interface. Any hardware access from the operating system or software running within each virtual machine, such as the use of CPU time, allocation of memory, access to storage or networking, is mapped to the underlying physical hardware.

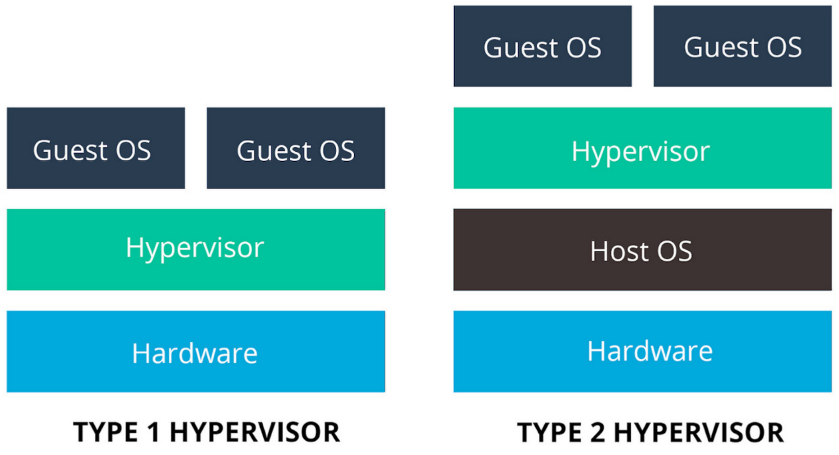

Figure 2.1: The two hypervisor types

As illustrated in Figure 2.1 (source: the Medium website[1]), two types of hypervisor are typically distinguished: Type 1 and Type 2 hypervisors. Type 2 hypervisors run on top of a separate operating system. The open source hypervisor VirtualBox[2] is an example. Users install VirtualBox on their normal computer operating system (such as Windows 10, Linux or MacOS), and can then create virtual machines running whatever (same or other) operating system is required. An advantage of Type 2 hypervisors is that they do not require a separate computer dedicated to the hypervisor.

[1]Type 1 and Type 2 Hypervisors: What Makes Them Different, `https://medium.com/teamresellerclub/type-1-and-type-2-hypervisors-what-makes-them-different-6a1755d6ae2c`

[2]`https://www.virtualbox.org/`

For example, a software developer can install VirtualBox on their normal development machine, alongside with their normal development environment, to try out the software under development under different operating systems in different virtual machines.

A significant disadvantage of Type 2 hypervisors is that they do not have direct access to the underlying hardware - they are managed by the underlying operating system themselves. Therefore, for high performance requirements, Type 1 hypervisors are better suited. Type 1 hypervisors such as the Citrix Hypervisor (formerly XenServer[3]) run directly on the physical hardware, without the use of a separate operating system. Therefore, Type 1 hypervisors have full control of the hardware, which generally allows them to achieve higher levels of availability, security and performance.

Modern virtualization software offers many powerful features to achieve these properties for their virtualized systems. It is beyond the scope of this book to explore them all in detail, or even to examine virtualization in more depth[4]. Instead, we will discuss just one advanced feature that is of particular importance in cloud computing: Live migration of virtual machines.

A data center that uses virtualization, especially a cloud data center, does not just use virtualization on a single physical server, but instead has a large number of physical servers running virtual machines on the chosen hypervisor. Live migration (sometimes also called hot migration) of virtual machines describes the ability to move a virtual machine from one physical server to another physical server with practically no interruption of the operating system and other software running in the virtual machine. For example, assume that the web server of an online retailer runs in a virtual machine on a specific physical server. There are many reasons why it might be desirable or necessary to move the virtual machine to a different server - for example, the server might be decommissioned to be replaced by a newer, more efficient model. If this is the case, an administrator would trigger the live migration of the running virtual machine to the hypervisor on a different physical machine. While this overall migration would take several minutes or even hours to complete, there would only be an extremely short period of time (probably a fraction of a second) during which the web server in the virtual would not respond as usual.

Live migration is performed in multiple steps:

1. The hypervisor on the source server contacts the hypervisor on the

[3]`https://xenserver.org/`

[4]Interested reader might refer to e.g. "Virtualization Essentials" by Matthew Portnoy

destination server and creates a new virtual machine with the settings of the source virtual machine.

2. The source hypervisor transfers the secondary storage of the virtual machine to the destination virtual machine.

3. The source hypervisor copies all read-only pages in memory to the destination VM.

4. The source hypervisor copies all read/write pages to the destination VM, and marks them as "clean" (i.e. already moved).

5. While the previous step happened, since the source virtual machine is running uninterrupted, it is likely that some of the read/write pages have been changed (e.g. because a customer put a product into their shopping cart in the meantime). Therefore, step 4 is repeated.

6. Note that step 5 should have taken much less time than step 4, because the number of "dirty" (i.e. modified) pages that have been changed during step 4 is likely rather small. If necessary, step 5 is repeated until the number of dirty pages becomes very small.

7. Once only a very small number of pages is "dirty", the source virtual machine is frozen, and the remaining dirty pages are transferred to the destination virtual machine.

8. Finally, it has to be ensured that all current users of the virtual machine are automatically (and transparently) switched over to the new VM instance on the new physical server. In typical configurations in which users connect through a proxy or load balancer, this can be realized by simply changing a destination IP address. Now the target VM is started, and the source VM can be turned off.

2.2.2 Virtualization Guarantees

Modern virtualization software offers the following three guarantees:

- **Isolation:** A virtual machine is strictly isolated from the underlying physical machine and any other virtual machines that might run on the same physical server. This isolation is crucial for security - a software bug or malware infection in the software running in a virtual machine does not put other virtual machines on the same physical server or the software running directly on the physical server at risk.

This isolation even extends to system performance, because modern virtualization software can limit the allocation of hardware resources such as memory, CPU time or network bandwidth to virtual machines. In other words, a virtual machine cannot use up so many system resources that it slows down other virtual machines on the same server.

- **Encapsulation:** An entire virtual machine (including the installed operating system and applications with all their data, the contents of main memory and the state of all virtual devices) is contained in a single file. This has multiple benefits. For example, the state of a virtual machine at a given point in time can be captured easily by copying that file, which allows the restoration of the virtual machine to its state at that point in time, as well as the creation of an identical clone of a virtual machine on a different copy of the hypervisor. Similarly, pre-configured applications can easily be shared in the form of a virtual machine export.

- **Compatibility:** The virtualization layer isolates the virtual machines (and all the software that runs in them - operating system and applications) - from the underlying hardware. This allows users to run software that would otherwise require legacy hardware on modern servers, as long as the hypervisor supports the virtualization of the legacy hardware. For example, a company might still have an application that requires an obsolete version of Unix that is no longer supported by current hardware. This application can be run in a virtual machine if the hypervisor supports the required legacy hardware as virtual hardware. Additionally, changes to the underlying physical servers do not affect the software installed in virtual machines, as long as the hypervisor is compatible with the new server hardware, because it would continue to expose the same unchanged virtual hardware to the virtual machines.

It should be pointed out that while these virtualization guarantees are generally met by modern hypervisors, significant configuration errors or software bugs (in the hypervisor) or even hardware bugs (e.g. in the CPU) can break them. The most famous example for the latter are the Meltdown and Spectre vulnerabilities[5] that affect most current server CPUs, and that allow sophisticated attacks that can circumvent the strong isolation guar-

[5] `https://meltdownattack.com/`

antees of hypervisors, allowing attackers to gain access to memory outside of their own virtual machine.

2.2.3 Benefits of Virtualization in Cloud Computing

Even though both hardware and software vendors have come up with mitigations for side-channel attacks like Meltdown and Spectre, a certain level of vulnerability remains. However, for practical purposes, if a malicious user manages to use the fairly sophisticated attacks to break out of their virtual machine and successfully accesses memory that belongs to a different virtual machine on the same physical server, not too much is generally gained, because users have no control over which other tenants they share the same physical server with. Therefore, in practice, other kinds of attacks are generally more promising, making vulnerabilities like Meltdown and Spectre not very relevant. Hence, the isolation guarantee between different virtual machines on the same physical server can generally be considered to still apply.

This directly leads to the first benefit of virtualization for public cloud service providers: It allows them to use multitenancy, i.e. to put the virtual machines of several different customers on the same physical server. Doing so lets public cloud service providers maximize their hardware utilization, because “left over” hardware resources on a physical server can be used for virtual machines of any other customer. In other words, there is no need to reserve physical resources for individual customers.

The fact that the state of a virtual machine is completely encapsulated in a single file makes it significantly easier for cloud service providers to achieve fault tolerance. Encapsulation (especially in combination with VM live migration) makes it easy to take snapshots of virtual machines that can be used as backups of VM state at a given time, or to move a virtual machine from a server with hardware issues (e.g. the failure of a disk in a RAID array) to a different server. The ability to move virtual machines between different physical hosts also allows cloud service providers to maximize the utilization of their servers. One frequently taken approach is to overprovision hardware resources. For example, a provider might choose to allocate more total memory to the virtual machines on a physical server than the server actually possesses - say, run ten virtual machines with a promised allocation of 8 GBytes of memory each on a server with only 64 GBytes. Most of the time, not all virtual machines will actually utilize all their allocated memory. If the required memory ever exceeds the physically available amount, some of the more memory intensive virtual machines can easily be moved to a different physical server with spare capacity, without the user even noticing.

Finally, compatibility of virtual machines allows cloud service providers to run very homogeneous data centers, even though their customers might require heterogeneous virtual machines. For example, customers might want to run applications on a wide range of operating systems, including different versions of Windows Server, several Linux variants and versions, and numerous other Unix systems, many of which might require legacy hardware. Rather than having to maintain a diverse collection of different servers supporting these operating systems and applications, virtualization allows cloud service providers to host all these different systems on hypervisors that run on whatever modern server platform is deemed to be most cost effective. As will be discussed in more detail in the next section, this homogeneity significantly reduces complexity and costs for the provider.

2.3 Modern Data Centers

This section gives an overview of what cloud computing data centers look like (see Figure 2.2, source: google website [6]), whether they are operated by organizations that choose to operate their own private cloud, or by public cloud service providers.

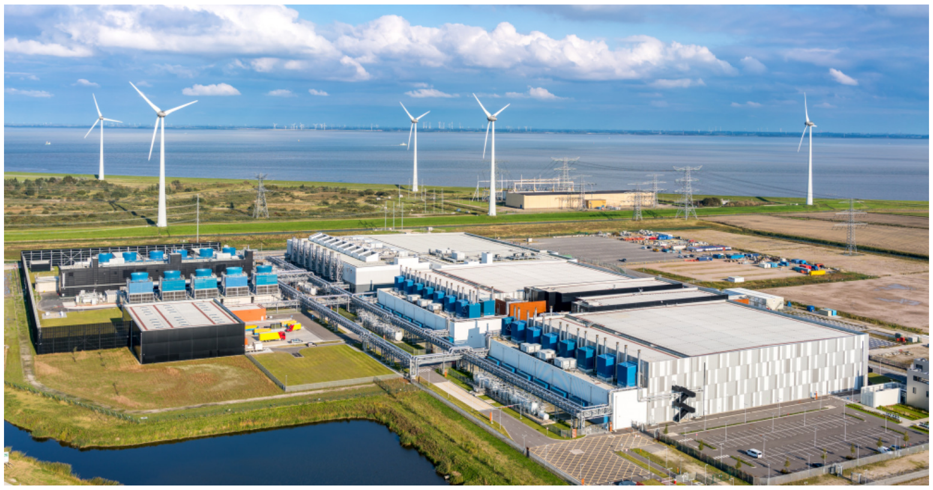

Figure 2.2: Google's data center campus in Eemshaven, Netherlands

[6]Google.com, `https://www.google.com/about/datacenters/locations/eemshaven/`

2.3.1 Data Center Design

Designers of modern data centers as they are used in cloud computing have a range of important aspects to consider, including:

- **Physical security:** Physical security is the most basic level of IT security. Whatever other measures are taken, if an unauthorized person gains physical access to a data center, security (confidentiality and integrity of data as well as the availability of applications) can no longer be guaranteed. Consequently, data centers have strong physical security features, including significant video surveillance and other intrusion detection systems, strong access control restrictions (often using biometric identification to ensure that only authorized individuals gain access) etc.

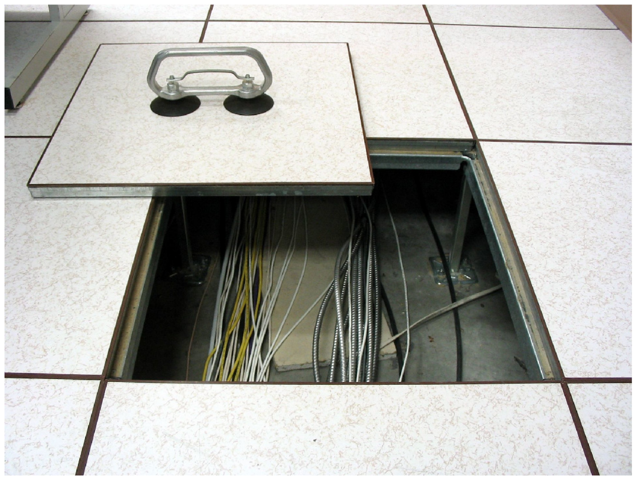

Figure 2.3: A raised floor creates a hidden void for the passage of cables.

- **The potential use of raised floors:** Another commonly used feature in data centers is that of raised floors (see Figure 2.3, source: Wikipedia [7]). Rather than putting server racks directly on the concrete floor, they are put on an elevated floor that is typically a few feet above the actual floor of the building. This has several advantages, including that cables connections can be run underneath the floor. Also, it can help with airflow for cooling the equipment, and it offers an additional line of defense against flooding events. Because water pools to the lowest

[7] `https://en.wikipedia.org/wiki/Raised_floor`

point, servers on the elevated floor will typically stay dry, while the isolated cables underneath are not affected by water. Some modern data centers do not use raised floors, for example if the room is not high enough to support them, or because alternative approaches to manage the airflow are used. As a final point with regard to floors (whether raised or not), attention has to be paid to the rather significant weight of tightly packed server racks - especially in buildings that are not specifically constructed as data centers, the floor might not be designed to hold the required weight safely.

- **Power:** Ensuring an uninterrupted supply of power is another major concern for large data centers. If possible, at least two redundant sources of power should be used (e.g. two different electricity providers, or at least two separate cables to connect the data center to the grid), to prevent the risk of a single fallen tree (or accidentally dug up underground cable) cutting power to the data center. In addition to that, power generators (typically powered by diesel fuel) are used in the event of power failures. Since it usually takes several minutes to start up such power generators to full power, the time until then is generally covered by uninterruptible power supplies, which are essentially big battery packs that have enough capacity to supply the connected equipment for a relatively short time period (e.g. 15 minutes).

- **Climate control:** Cooling a data center is another major concern. Most of the electricity used by servers and other components such as networking equipment is ultimately transformed into heat. Servers are usually tightly packed to reduce space requirements, which in larger data centers means at least tens of thousands of servers in a relatively small floor space. Since each server might create a few hundred Watts of heat, the total cooling requirement of a larger data center can easily be in the range of Megawatts. In most climates zones, this type of cooling requires significant air conditioning equipment, and careful planning of fan placement to optimize air flows. To increase heat transfer and efficiency, water cooling approaches can be used. In addition to cooling the data center, climate control also has to ensure an adequate level of humidity. If humidity is too high, metal contacts running electricity might corrode over time, while very low levels of humidity might lead to the build-up of electric charges on different components that, once they discharge, can also damage equipment.

- **Network connection:** Data centers, especially cloud data centers, clearly depend on high bandwidth network connectivity that is very reliable. For that end, similar to the supply of power, multiple providers with separate physical connections are generally desirable. Different approaches are possible - either normal operation uses more than one network provider, or it uses only one with alternatives available on standby in the event of outages.

- **Fire suppression systems:** Modern buildings generally have fire detection systems with sensors for smoke or heat that trigger sprinklers to extinguish fires with water before they get out of control. This is obviously not a good solution in a data center, because water spray is likely to damage computer equipment due to short circuits caused. Consequently, data centers are usually equipped with fire suppression systems that work by reducing the oxygen levels in the room by replacing it with other gases such as argon.

- **Server (and networking equipment) organization:** In order to maximize the utilization of space, modern data centers organize servers in rows of racks, each of which contains multiple servers stacked on top of each other (see Figure 2.4, source: Tradeindia website [8]). The exact organization, including aspects such as the placement of networking equipment (switches and routers), cabling and cooling equipment is a complex optimization problem for which different approaches exist. For example, cables can be run under a raised floor or above the equipment on the ceiling and power equipment such as uninterruptible power supplies can be more centralized or distributed.

2.3.2 Modern Server Design

In the past, servers often used different hardware architectures than contemporary desktop or laptop machines. While there are still some alternatives in the marketplace such as ARM-based servers, most servers today use CPUs by Intel or AMD based on the same x86-64 instruction set as current consumer computers.

Despite this, servers used in data centers of course have significant differences compared to for instance desktop PCs. To begin with, different form factors are usually used, which are optimized to allow efficient stacking of servers in server racks. Additionally, servers are of course usually

[8] `https://www.tradeindia.com`

Figure 2.4: A server rack is a structure that is designed specifically to house technical equipment including routers, switches, hubs, and of course, servers.

rather powerful, with often multiple CPUs with multiple CPU cores per server, large amounts of memory, secondary storage and high performance networking capabilities.

Additionally, servers often have features that increase reliability and fault-tolerance. This typically includes redundant power supplies and hard disks (e.g. in RAID arrays) that are hot-swappable. In other words, if for instance the power supply of a server fails, the redundant secondary power supply takes over instantaneously, and an alarm is raised. This allows maintenance technicians to replace the faulty power supply without ever turning off the server. Another common feature is *Error-Correcting Code* (ECC) memory that can detect and correct random bit errors in main memory.

Finally, servers often come with advanced management features. Server mainboards usually contain a baseboard management controller that implements the *Intelligent Platform Management Interface* (IPMI). This interface allows remote access to servers, for example to install an operating system or to reboot a hanging server, independently from any installed operating system. While software with similar features exists that can be run under common server operating systems, such software depends on the server being responsive, while the IPMI still works even if the system is down or if no operating system has been installed yet.

2.3.3 Modern Storage Approaches

Data centers have used different approaches to organizing secondary storage over time. Traditionally, each server stored the data it used on local hard disks. To increase reliability and performance, these disks were typically organized in a *Redundant Array of Independent Disk* (RAID) array. A significant downside of this approach is that the exact storage capacity each individual server in a data center needs is hard to predict and might vary over time. To prevent running out of local storage, disk capacity would have to be assigned generously, resulting in relatively low average space utilization, i.e. significant amounts of unused disk space.

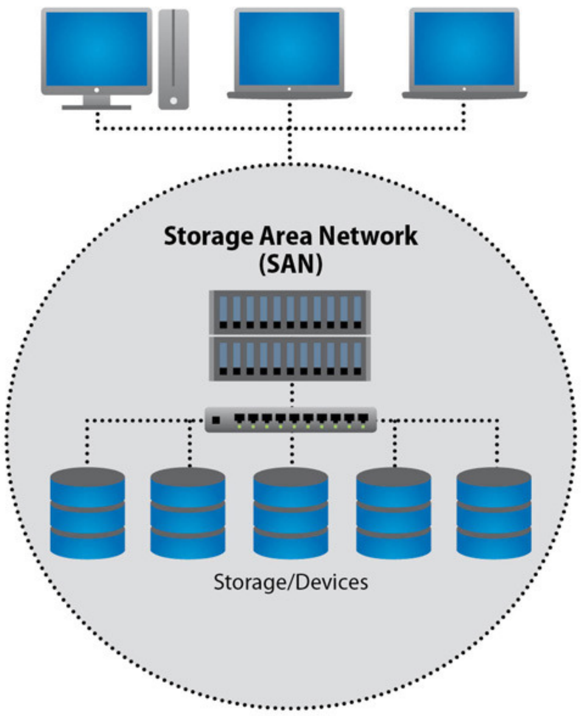

Figure 2.5: A storage area network.

Consequently, secondary storage for a large number of servers was often consolidated in *Storage Area Networks* (SAN) (see Figure 2.5, from 10GTek website[9]). In a SAN setup, a large number of disks is combined in a central-

[9]The 10Gtek website, `https://www.10gtek.com/new-1419`

ized storage array. The servers are connected to the storage array through a high-speed network such as Fibre Channel that is separate from the regular local area network through which all other network traffic runs. From the perspective of the server, storage in the SAN can be accessed as if it was local disk storage. The big advantage of this setup is that storage can flexibly be assigned to servers, resulting in higher utilization and less unused storage capacity.

While SANs are still widely used in practice, a different approach that moves storage back to the individual servers is increasingly adopted. There are several reasons for this, including the fact that SANs tend to be expensive, and that regular Local Area Networking (LAN) speeds have become fast enough to allow servers to efficiently share their local storage with other servers. In these new approaches that are often referred to as software defined storage, the disks installed locally at each server are combined into a large virtual pool of disks, the storage capacity of which is then flexibly assigned to individual servers. This general approach in which storage, networking and compute resources are no longer separated, but combined back into the individual servers making up the data center and virtualized is also referred to as Hyperconverged Infrastructure.

With regard to the actual storage devices used, flash memory based *Solid State Disks* (SSDs) that offer significantly lower access times and a much higher number of *input and output operations per second* (IOPS) than traditional magnetic hard disks are increasingly being used. While their cost is still higher, the difference has become small enough to make the use of SSDs for frequently accessed or performance critical data such as installed operating systems and programs, relational database systems etc. worthwhile. For other kinds of data, especially where access times are less important, hard disks are still used. For example, for large image or video files sustained data transfer rates are generally more important than access times, making magnetic hard disks the more economic choice. Even magnetic tapes (usually used in large tape libraries, in which automatically operated robot arms retrieve and insert the required tapes into one of several tape drives) are still used and economical for data that is kept for mostly archival purposes, and very rarely accessed, because the per-Gigabyte cost of tape storage is even lower than that of magnetic hard disks.

2.3.4 Cost and Other Considerations

Modern cloud data centers, especially those operated by public cloud providers, tend to be very large, often housing at least tens of thousands of servers.

One of the main reasons for this is that the many of the costs associated with running a data center scale much slower than linearly with the numbers of servers. Outside of volume discounts that cloud providers receive for their hardware purchases, the actual cost of purchased computing, storage and networking equipment is one of the few factors that scales relatively linearly. Many other costs - e.g. the cost for physical security and access control measures, technical measures such as redundant power systems, fire suppression systems etc. do increase with the size of the data center, but much slower than linearly. For example, if a handful of security guards can secure a small data center of one hundred servers around the clock, a few more personnel are likely required to secure a data center with a hundred times as many servers - but not nearly a hundred times as many.

In order to keep costs low, operators of large data centers place a very high value on the automation of processes. For example, if the installation of the software on new servers was done manually, the effort required would increase proportionally to the number of servers. If software installation (and other routine tasks) are automated through a set of scripts, the number of system administrators required to operate say a thousand servers is not much higher than the number needed to operate a hundred servers.

Another important consideration is the choice of location for data centers. While performance considerations (especially network latency) dictate that data centers are located relatively close to their intended end users, their precise placement depends on other factors, such as the risk of natural disasters at different locations. Proximity to active volcanoes, geological fault lines or even low lying locations with high flooding risk should generally be avoided. Furthermore, factors like climate (colder is generally better, to make cooling easier and cheaper) and the cost of electricity play a very important role.

Since disasters (natural and otherwise) can never be entirely prevented or mitigated, disaster recovery approaches are another important consideration. The fact that public cloud service providers operate a number of different data centers across the globe helps with that: If a data center experiences a significant outage, others can take over the load, and while that might result in some performance degradation for users, this is much preferable to a complete outage. Another related aspect is that backups of data should be synchronized between data centers in different locations to minimize the risk of data loss.

A final consideration is the use of proprietary vs. open source technology. This is not only relevant for software, but also for hardware (see e.g.

the *Open Compute Project* (OCP)[10]). One of the reasons for the use of open source technology is potential cost savings, because licence costs typically are incurred for each individual machine on which e.g. a software package is used. In other words, while the cost of a $100 software package on an individual PC might not seem excessive, buying ten thousand licences for such a package to use on all the servers in a large data center costs a million dollars, making the in-house development of custom software an often attractive option. Such software might be made open source, to encourage other users to adopt it and to eventually contribute improvements from which the original developer can benefit as well, or be kept as in-house proprietary software if it is deemed to deliver the developing organization a competitive advantage that it does not want to share with others.

2.4 Summary

The large success of modern cloud computing has been made possible by relatively recent technical progress in key areas such as networking and virtualization. While cloud computing users generally do not need to understand the technical details of the underlying technologies, a basic understanding of them is crucial for anyone who wants to design secure, cost-effective and reliable applications in the cloud.

[10] `https://www.opencompute.org/`

Chapter 3

Cloud Computing Platforms

In this chapter, we discuss common characteristics, features to compare and the architectures for both commercial and open source cloud platforms. The list of the cloud platforms available is changing day-by-day due to the dynamic nature of the field. Consequently, while we discuss the most active projects at the time of writing, this can have changed by the time this is read.

3.1 Introduction

Over the last few years, many different cloud platforms have been proposed both as commercial products and as open source projects. In Table 3.1, we list the most important cloud platforms at the time of writing. In the rest of this chapter, we illustrate the features to compare between the various platforms, the standard architecture and finally we describe in detail the most important cloud platforms.

3.2 Features to Compare

There are several features that must be considered when a company has to choose a cloud platform to run its business. Each of the platforms has different characteristics of the same feature, the same feature can be provided differently from one cloud platform to another or it can have different costs. For these reasons, the comparison between the various solutions could be a difficult task. In this section we list the main features to take into consideration when a company decides to adopt a cloud computing platform in

Name	**Website**	**Note**
OpenStack	`www.openstack.org`	Open source
phoenixNAP	`phoenixnap.com`	-
Kamatera	`www.kamatera`	30 day free trial
Amazon Web Services	`aws.amazon.com`	Where everything starts
Microsoft Azure	`azure.microsoft.com`	Microsoft office in the cloud
Google Cloud Platform	`console.cloud.google.com`	Focus on the PaaS
Adobe	`www.adobe.com/creativecloud`	20+ for adobe apps in the cloud
VMware	`www.vmware.com`	From hypervisor to the cloud layer
Dropbox	`www.dropbox.com`	The first popular cloud storage
IBM Cloud	`www.ibm.com/cloud`	-
OpenNebula	`www.opennebula.io`	Open source, multi tenancy
Rackspace	`www.rackspace.com`	OpenStack with professional support
Red Hat Cloud suite	`www.redhat.com`	As for Rackspace
Saleforce	`www.salesforce.com`	The first popular Customer Relationship Management (CRM) on the cloud
Egnyte	`www.egnyte.com`	-
Navsite	`www.navisite.com`	Combine the best service from Azure, AWS and VMWare
Oracle Cloud	`www.oracle.com`	-
SAP	`www.sap.com`	-
Verizon Cloud	`www.verizon.com`	Cloud service and backup for your phone

Table 3.1: Some of the most active cloud platforms.

order to improve its business.

3.2.1 Application Programming Interface (API)

Most cloud platforms provide a web portal where users can interact with the offered services such as starting/shutting down virtual machines, managing storage, monitoring resource usage, configuring load balancers etc. However, some services provided by cloud platforms are not available via a web portal, so it is important that the cloud platform exposes the API to interact with all the features it provides. Moreover thanks to API support, a developers or a Cloud Manager can write routines to fully exploit the features available and at the same time automate procedures in order to improve the application performance or to prevent/recover from failures.

3.2.2 Availability Zones

At 12:47 AM PDT on April 21, 2011 an invalid traffic shift prior to a network upgrade caused some Amazon cloud services to lose connectivity in a wide area located in north-east USA[1]. The services were fully recovered only after 3 days, and the outage affected big name Amazon cloud customers, including popular Web sites like Foursquare, HootSuite, Quora and Reddit or applications like Twitter. For some of them, this outage meant significant financial losses that AWS reimburses with just a "10 day credits" of the services used by the company involved in the outage. In a note after the outage, AWS declared that it will work to make it easier for customers to take advantage of multiple Availability Zones.

The example above points out the importance of having different availability zones and of using them to improve both performance and reliability. In particular, by having different availability zones, the computation can be close to where the output data are necessary (this can improve the application performance) or more replicas of the same data/application can be spread in different locations so an outage in a specific area cannot prevent access to the data/applications.

3.2.3 Fault Tolerance and Failover

One of the informal definitions of cloud computing is: "Cloud computing simply increases the number of things that can go wrong. And go wrong they do." Applications provided by a cloud computing platform could fail

[1] `https://aws.amazon.com/message/65648/`

in many ways due to hardware, software or network problems and they usually have to deal with a huge number of users. For these reasons it is necessary to have some fault tolerance or failover mechanisms. Some cloud platforms provide such mechanisms as part of the service available, so it is important to consider them as an essential feature.

3.2.4 Migration

Most of the applications in cloud platforms run inside a virtual machine. During its lifetime, it could be necessary to move the virtual machine from one physical machine to another, for example to another availability zone by considering the network traffic. Usually, cloud platforms provide two types of migration: the "hot" migration (also called live migration) where the service running inside the virtual machine is provided without any interruption or "cold migration" where the service has to be stopped for a while: the time necessary for another virtual machine to start the service and continue to provide it to the users connected to the previous virtual machine. This operation involves many sub tasks related to process management, session management and, of course, networking management so it is really important that this migration is fully supported by the cloud platform, so the developer can focus on the application and not on the migration procedure.

3.2.5 Monitoring

Any cloud platform has some kind of monitoring tools to detect when a VM is ready, failed, booting, etc, but what is really interesting for a cloud user is what measures are available and with which granularity. The measures can be cpu usage (%), memory or disk occupancy (bytes), network traffic (bps) just to name a few, while the granularity is the frequency of the sample taken: for example, it could be one sample per second, one sample per minute or one sample per hour. Moreover, it is important to figure out if these measures are reachable by API so a developer can use these measures inside some routine. The monitoring capacities significantly differ from one platform to another in terms of both variety and granularity.

3.2.6 Cloud Federation and Open Virtualization Format

Despite the efforts from the *Open Cloud Computing Interface* (OCCI) open community, the *Open Cloud Consortium* (OCC) and the supporters of the

Open Virtualization Format (OVF) standard, every cloud computing platform has selected its own protocol, its own API and its own command-line interface to interact with the platform itself. Concerning the command-line interface, Figure 3.1 shows how to perform the same action (that is, starting up a virtual machine) in AWS (see Fig. 3.1(a)), in OpenStack (see Fig. 3.1(b)) and in GCP (see Fig. 3.1(c)). It is clear how the same action can

```
$ aws ec2 run-instances \
    --image-id ami-1a2b3c4d \
    --count 1 \
    --instance-type c3.large \
    --key-name MyKeyPair \
    --security-groups MySecurityGroup
```

(a)

```
$ openstack server create --flavor 1 --image 397e713c-b95b-4186-ad46-6126863ea0a9 \
  --security-group default --key-name KeyPair01 --user-data cloudinit.file \
  myCirrosServer
```

(b)

```
$ gcloud compute instances create "my-new-instance" \
      --zone="us-west1-b" \
      --image-family="tf-latest-cu92" \
      --image-project=deeplearning-platform-release \
      --maintenance-policy=TERMINATE \
      --accelerator="type=nvidia-tesla-v100,count=8" \
      --machine-type="n1-standard-8" \
      --boot-disk-size=120GB \
      --metadata="install-nvidia-driver=True"
```

(c)

Figure 3.1: How to run a new VM by using the command-line client software provided by (a) AWS, (b) OpenStack and (c) GCP respectively.

differ from one platform to another in terms of parameter names, sequence and cardinality. This situation is called "lock-in" (already discussed in Section 1.3.2): with the exception of a few attempts made by no longer active cloud platform projects like *Nimbus*[2], all cloud platforms have defined their API/library/command-line interfaces and they avoid compatibility with other cloud platforms on purpose. This way, a user/developer who starts to get confidence with a cloud platform is discouraged from learning how to use a different platform. This is even more complicated if the developer works for a company who needs to have their services in the cloud as soon as possible. In the recent years, some software tools capable of interacting with different cloud platforms have been introduced. Some of these tool are discussed in Chapter 7.

[2]`https://www.anl.gov/mcs/nimbus-cloud-computing-for-science`

3.2.7 Elasticity

One of the advantages of the cloud computing is elasticity, i.e. the ability to dynamically adapt the number and type of resources used depending on workload fluctuations. In order to realize this elasticity, it is necessary to predict the incoming workload and the reaction-time, i.e. how fast the system is able to start up/shut down the resources dedicated to a particular workload. Depending on how accurate the monitoring tool and on how reactive the system is, one cloud platform can perform much better than another in response to workload fluctuations.

3.2.8 User Management

A cloud platform can be used by various types of users and developers with different skills and different responsibilities. Therefore, it is important that a cloud platform can easily create different types of users with a different visibility of the whole platform. In particular, this means that a user should have a list of actions allowed, and all other actions should be forbidden. The permissions related to a user or to a group of users should not only be easy to set but also easily changeable at run time.

3.2.9 Services provided

Nowadays cloud platforms are not only computational resource providers but also services providers. The services available in each cloud platform (in particular the commercial ones) are numerous and very different from each other. Concerning the quantity of services available, at the time of writing, AWS and Azure declare more than 200 services, while GCP provides around 100 services, just to cite the three main cloud providers. The web consoles of the cloud providers divide these services into different categories, for example *Databases* (where the cloud user can exploit different solutions such as NoSQL databases, relational database services, in-memory cache service, just to name a few), *Machine learning* (where the cloud user can find services able to provide different Deep Learning models, time-series forecasting, Natural Language Understating etc.) or *Containers* (where the cloud user can manage a Kubernetes installation very easily).

3.3 Architecture

As mentioned before, there are several cloud platforms with various features and/or services available. Despite these differences, the core components of each cloud platform are similar. In Figure 3.2, we propose a general cloud platform architecture that, to the best of our knowledge, can fit any cloud computing platform mentioned in Section 3.1.

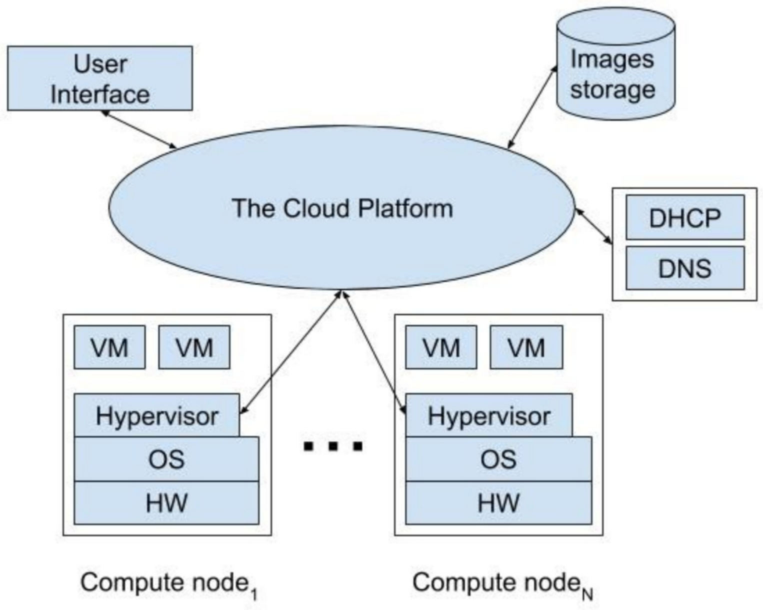

Figure 3.2: The common cloud platform architecture.

The compute node is where the virtual machine (or containers) run to provide the services required. Inside a compute node, we can find the hardware, the operating system, the hypervisor and, finally, the virtual machines. Any of these components are necessary and any of them can be compatible or incompatible depending on the chosen cloud platform. In particular, if a system administrator plans to install a cloud platform, they for example have to verify which operating systems are compatible with the cloud solutions proposed.

DNS and DHCP are fundamental components of any cloud solution since any service provided by cloud computing is a network application. This means that it needs at least an IP address where the user can send their requests. Moreover the VM in a cloud environment can be started or shut down any time and with a high frequency. Therefore, these components must be able to assign an IP address to a VM and make it reachable from everywhere in the world in a fast and efficient manner.

The images used by virtual machines contain operating system files, libraries specific to a service and, in some cases, also user data. This means that an image can occupy several Gigabytes of disk space, and if hundreds or thousands of images are stored, image storage alone can become a challenge. As a matter of fact, any cloud platform has a specific component in charge of managing virtual machine images. These images not only have to be stored, but they also must be moved from the image repository to compute nodes. Even when they are not claimed by any service, they can be moved/replicated from an image repository to another for performance or fault tolerance purposes. For example, if a very important sport match (the Super Bowl or the UEFA championship final) will be seen by million of fans using a cloud service in a specific country, the cloud platform can decide to in advance move the image of the VM in charge of providing the streaming of the match into the availability zone closest to where the highest number of fans (and, in this case, also the highest number of users of the cloud service) is expected.

Last but not least, the user interface is really important. The major public cloud platforms provide several different interfaces: a web interface, a command-line interface and libraries for various programming languages.

3.4 Commercial and Open Source Cloud Platforms

This section presents the most important cloud platforms at the time of writing. As mentioned in Section 3.1 there are many different platforms and more will probably enter the market, but in the last few years the top four providers have been AWS, Microsoft Azure, Google Cloud and Alibaba Cloud as indicated by Canlys[3].

3.4.1 Amazon Web Services (AWS)

The introduction by Amazon of its Elastic Compute Cloud (EC2) service in 2006 marked the true beginning of modern cloud computing. The roots for the idea of AWS go back to the 2000 timeframe when Amazon was a different company than it is today: a simple e-commerce company with a scalability issue. In order to face this problem, the company decided to

[3] `https://www.canalys.com/newsroom/worldwide-cloud-infrastructure-services-Q1-2020`

build some solid internal systems to manage the hyper growth it was experiencing. At the beginning these core systems were used just to manage the e-commerce division of the company, and only a few years later Amazon started to plan how to make a business out of these systems[4].

Moreover, in 2000, the company also wanted to help third-party merchants (such as Target, Marks & Spencer just to name a few) to build online shopping sites on the top of Amazon's e-commerce platform. This was the first step toward what would become AWS, since the developers at Amazon started to create well-documented APIs to make it easy for third parties to use the Amazon e-commerce engine to sell their products. In just three months, Amazon developers built the API, database, compute and storage components of the system with no thought to scale or reuse. The above components were used by different teams within Amazon, and developers started to think about these components as a set of common infrastructure services everyone could access. This is when Andy Jassy, who was Amazon CEO Jeff Bezos' chief of the staff at the time, began to realize they might have something bigger. In 2003, three years before the launch of AWS, by providing infrastructure services (like compute, storage and database) to run what was becoming the biggest e-commerce website in the world, Jassy and the developer teams realized that they achieved high skills at running reliable, scalable and cost-effective data centers. During the fall of 2003, Jassy started to think about the Internet as an "operating system" on top of which the company could build their services, instead of building applications from scratch. In August 2006, Amazon launched "Amazon Elastic Compute Cloud" and surprisingly, for several years, no competitors responded. At the time of writing, AWS

- comprises more than 212 services besides the most famous ones like Elastic Compute Cloud (EC2) and Simple Storage Service (S3). Some services cannot directly be used by end users but are offered through APIs for developers for their applications;

- has distinct operations in 22 geographical "regions" world wide, and new regions are announced every year. Each regions has multiple "Availability zones" that are one or more data centers;

- has come to play a crucial role at Amazon as its most reliable source of income. In the first quarter 2021, AWS contributed $4.16 billion in

[4]https://techcrunch.com/2016/07/02/andy-jassys-brief-history-of-the-genesis-of-aws/

operating income, that is nearly 47% of Amazon's overall operating income[5];

- has notable customers like NASA and Netflix.

AWS is where the cloud computing started in 2006, and it is still the leader in the market, as is illustrated in Figure 3.3 (source: Business2community website [6]).

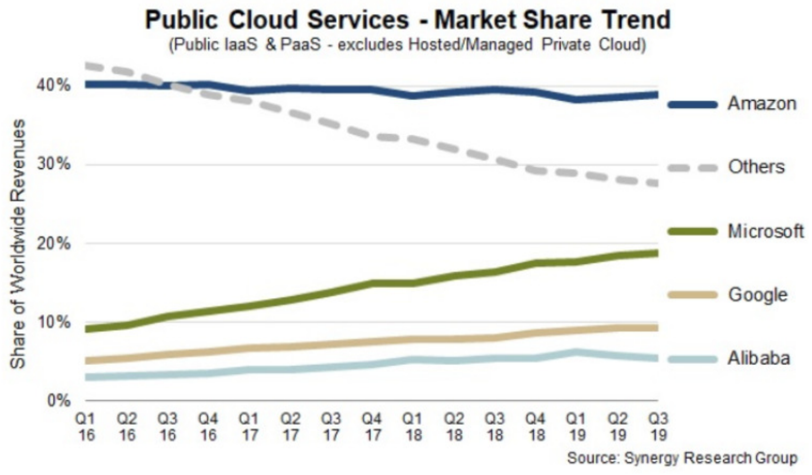

Figure 3.3: Public cloud services market share trend.

3.4.2 Microsoft Azure

Microsoft's VM service was announced as Windows Azure in 2008 and became commercially available in February 2010. In 2014 it was rebranded as Microsoft Azure and is now the second largest public cloud platform in the world after AWS.

Microsoft lists over 200 Azure services related to areas like computation, identity, mobile, storage, messaging and many others. At the time of writing, Azure is available in 54 regions around the world, and every year Microsoft announces new regions to be opened. From the developer's

[5]Source: `https://www.cnbc.com/2021/04/29/aws-earnings-q1-2021.html`

[6]`https://www.business2community.com/cloud-computing/the-latest-public-cloud-market-share-and-beyond-02258898`

point of view, Azure provides an API built on REST, HTTP, and XML used to interact with the services mentioned above.

One of the main differences between Azure and other cloud platforms is the deployment models provided. Besides the "classic" deployment model where each resource - like VM, database, etc. - is managed individually, Azure in 2014 introduced the Azure Resource Manager where users can create closely coupled groups of resources that can be deployed, managed and monitored together. While AWS has grown by being the developer-friendly platform, Azure has grown thanks to large organizations already committed to Microsoft products deciding to start using Microsoft cloud services.

3.4.3 Google Cloud Platform (GCP)

Two years later than AWS, Google launched its computing service in 2008. The service was called "App Engine" and initially was just a developer tool that allowed users to run their web applications on Google infrastructure. At the beginning, App Engine was made available to 10,000 developers that could use the service with some restrictions: 500 MB of storage, 200 million megacycles of CPU per day and 10 GB of bandwidth per day. Only in November 2011, Google made App Engine a public service and now Google provides what it is called "Google Cloud Platform" (GCP) with many different cloud services. Nowadays, several big companies have decided to use GCP to run their business (Airbnb, Zillow, Bloomberg and PayPal just to name a few).

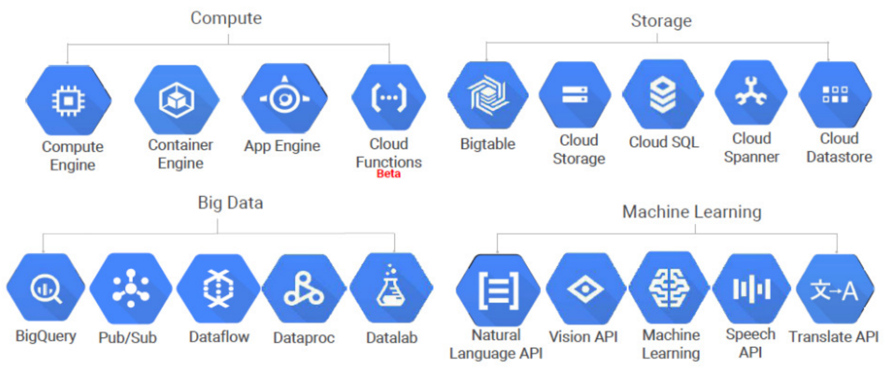

Figure 3.4: Google Cloud Platform Services

Figure 3.4 (source: Google cloud website [7]) illustrates a subset of the various GCP services (Google lists over 90 products under the Google Cloud brand). It is worth to mention that some of the services provided are the same as everyday google users use. For example, both the "Speech API" and "Natural Language API" are utilized every time we talk to our Android phone and Cloud Search is the service used anytime we perform a search in the Google search engine. At the time of writing, GCP is available in 22 regions and 61 zones.

3.4.4 Alibaba Cloud

Alibaba is the largest e-commerce company on earth with its three main e-commerce sites: Taobao, Tmall and Alibaba.com. At the time of writing, it has 674 million active users[8] and it continues to grow. As for Amazon, cloud computing initially was a platform to support its e-commerce websites for Alibaba. In September 2009, Alibaba Cloud was founded to provide cloud services to the public. In particular, Alibaba Cloud (also known as Aliyum) offers cloud services to online businesses such as Elastic Compute, Data Storage, Content Delivery Networks and so on. Currently, Alibaba Cloud operates in 21 regions and 63 availability zones.

3.4.5 OpenStack

The origin of OpenStack dates back to 2010 when two companies decided to work together to exploit their experiences and expertise. In particular, Rackspace wanted to rewrite the code related to its cloud infrastructure and, at the same time, Anso Labs (contracting for NASA) published the Python source code for its cloud computing fabric controller. The joint work of the developer teams formed the base for OpenStack which was officially announced at the Open Source Software Conference (OSCON) in Portland on July 21st, 2010[9].

The OpenStack mission is "to produce a ubiquitous Open Source cloud computing platform that is easy to use, simple to implement, interoperable between deployments, works well at all scales, and meets the needs of users and operators of both public and private clouds".

While OpenStack is not the only open source software able to create a Cloud Platform, it is clearly the most important one. Other projects (e.g.

[7] `https://cloud.google.com/`

[8] `https://www.thestreet.com/world/history-of-alibaba-15145103`

[9] `https://docs.openstack.org/project-team-guide/introduction.html`

Eucalyptus and Nimbus) were not able to keep their software up-to-date. One of the reasons for the success of OpenStack is the OpenStack Foundation: a non-profit corporation that promotes the project and its community. This foundation has been able to involve more than 500 companies like AT&T, Ericsson, Huawei, Intel and Red Hat, just to name a few.

Another reason for the success of OpenStack is related to its architecture and its project management. In particular, OpenStack is made up of various components (all of which play a role in managing the entire system). For each component, there is a developer team dedicated to the component, a mailing-list and a community. Like for the ISO/OSI stack, each component can improve its implementation independently from other components. The main components are illustrated in Figure 3.5 (source: OpenStack website[10]).

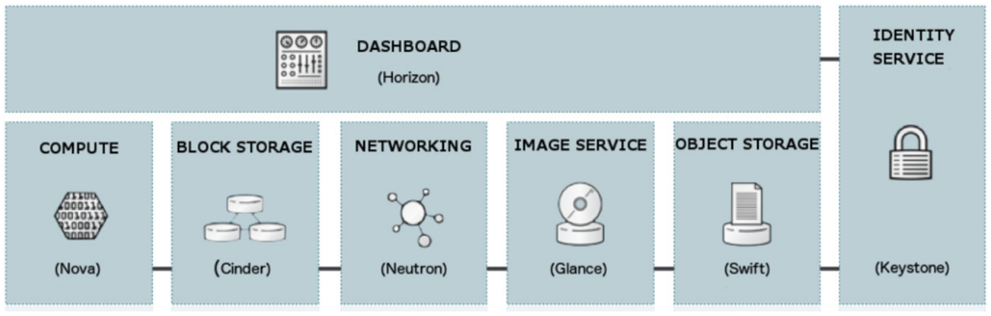

Figure 3.5: The main OpenStack components

Each component has a specific role[11]:

- **Horizon:** provides a single User Interface used for VM deployment, configuration and monitoring.
- **Nova:** compute service, controls compute resources and provides its telemetry.
- **Cinder:** block storage service, provides control over block storage resources (i.e. SSD).
- **Neutron:** networking service, provides control over networking (network abstraction layer).

[10]`https://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html`
[11]`https://www.tietoevry.com/en/blog/2020/07/openstack/`

- **Glance:** provides services and libraries to store and manage bootable disk images.
- **Swift:** object storage service, provides scalable object data storage.
- **Keystone:** authentication service, provides common authentication methods.

Since OpenStack is free, installations of this Cloud Platform that users can try out for free are available online. Some of the available websites are *Chameleon Project*[12] and *CloudLab*[13].

3.4.6 OpenNebula

OpenNebula is an open source project that provides a cloud computing platform for managing heterogeneous distributed data center infrastructures[14]. OpenNebula started in 2005 as an internal research project of the University of Madrid and it is the only European open source platform to build an Infrastructure as a Service (IaaS) platform.

It has been designed to be a simple but feature-rich, production-ready, customizable solution to build and manage enterprise clouds. OpenNebula combines existing virtualization technologies with advanced features for multi-tenancy, automated provisioning and elasticity. In particular, OpenNebula supports Xen, KVM and VMware hypervisors, and the Linux distributions Ubuntu and Red Hat Enterprise have already integrated this cloud platform.

As illustrated in Figure 3.6 (source: OpenNebula forum [15]), the OpenNebula deployment is similar to a classic cluster architecture. In particular we have the following main components:

- **Master node:** it is the front-end of the system, in charge of queuing, scheduling and submitting jobs to machines in the cluster. More specifically, it provides a user interface to create virtual machines and monitor their status.
- **Worker node:** these machines provide raw computing power for processing the jobs submitted to the cluster. More specifically, they deploy virtualisation supervisors, such as VMware, Xen or KVM.

[12]`https://www.chameleoncloud.org/`

[13]`https://www.cloudlab.us/`

[14]`https://opennebula.io`

[15]`https://www.opensourceforu.com/2017/02/an-introduction-to-opennebula/`

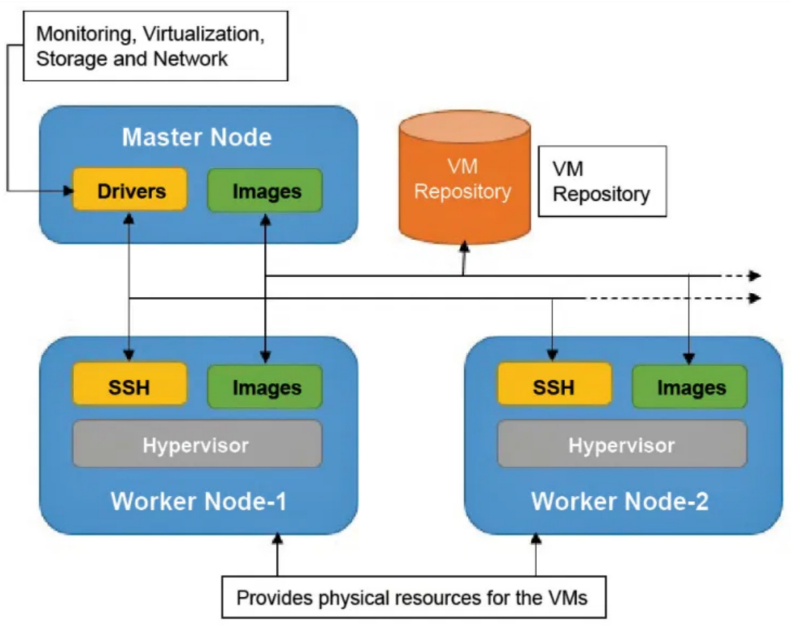

Figure 3.6: The OpenNebula Architecture

The project includes features for integration, management, scalability, security and accounting and it also claims standardization, interoperability and portability, providing cloud users and administrators with a choice of several cloud interfaces such as Amazon EC2, OGF Open cloud computing Interface and vCloud. It can also accommodate multiple hardware and software combinations in a data center. The success of OpenNebula is owed to OpenNebula systems[16]: an international software company that develops and provides commercial support for this cloud platform. Thanks to this company, OpenNebula is widely used in a variety of industries including cloud providers, telecommunications, government, banking, gaming and research laboratories. Notable users from the telecommunications and internet industry include Akamai, Blackberry, Telefónica and INdigital.

3.5 Summary

Since cloud computing is dominated by a small number of large public cloud computing providers like AWS, Azure and the Google Cloud plat-

[16]https://en.wikipedia.org/wiki/OpenNebula_Systems

form, it might seem that prospective cloud service users do not have a lot of choice when it comes to picking a cloud platform to use. This chapter has shown that this is not true. Not only is there a fairly diverse number of public cloud providers, but there are also several open source projects that allow users to create their own private cloud. Since the different providers and platform all have different strengths and weaknesses, it really depends on the specific needs of an organization which cloud platform is best suited for it.

Chapter 4

Types of Cloud Services

This chapter describes the wide range of services that modern clouds offer to their users. These services can be categorized broadly into three levels of abstraction, which are described in section 4.1. The rest of this chapter gives an overview of the types of cloud resources that are available in modern cloud systems, using the Google Cloud Platform as an example.

4.1 Levels of Abstractions

As described in the introduction, cloud services are typically categorized into three service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). To briefly recap, the cloud service provider offers bare (virtual) computing resources such as virtual machines or storage in IaaS, higher-level computing platforms in PaaS, and ready-to-use software applications in SaaS. An often used analogy to illustrate these different service models is shown in Figure 4.1 (source: A. Barron[1]).

Someone who wants to eat pizza has four different ways to achieve this outcome:

- **Made at home:** In this scenario, the dining table (including chairs, cutlery, drinks etc.), the oven (including its power source), the pizza dough and all the toppings - in short, everything - is in the responsibility of the pizza eater. This is equivalent to the traditional on-premise hosting of IT, where everything (physical servers including power supply and cooling, operating systems and configurations,

[1]The Pizza-as-a-Service metaphor was firstly introduced by Albert Barron in 2014.

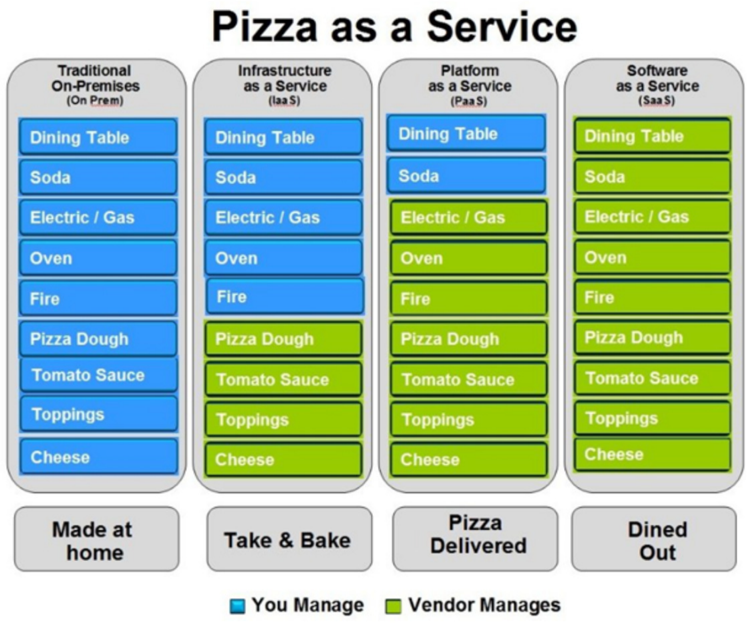

Figure 4.1: The Pizza-as-a-Service metaphor.

platform elements and the applications themselves) is in the responsibility of the user.

- **Take & Bake:** In this model, everything but the pizza itself remains in the responsibility of the user - i.e., they still need to have everything to actually bake and consume a ready-to-bake pizza. This can be compared to the IaaS model, where the majority of the responsibility still lies with the user (application software, platform software, operating system, configurations), but where the responsibility for maintaining the physical hardware rests with the cloud service provider.

- **Pizza Delivered:** Here, the user is only responsible for the eating environment, while the raw pizza and its baking (including everything that is needed for it) rests with the delivery service. The cloud analogy would be PaaS, where the user remains responsible for their software applications, but hands the physical hardware, its operating system and all required platform elements over to the cloud service provider.

- **Dining Out:** In this final model, the user simply has to consume the pizza - the responsibility for everything rests with the restaurant. The cloud equivalent is SaaS, where the cloud service provider offers a ready-to-use application, and takes care of everything that is required to run it.

How do you decide between these different models of pizza consumption? There is no best way to consume pizza - it depends on the circumstances and individual preferences. Broadly speaking, in the figure, both the price and the level of convenience increase from the left to the right. On the other hand, the level of control decreases from the left to the right: If you make and eat pizza at home, you can make whichever kind of pizza you want, and you can create the exact physical setting in which you want to consume the pizza. On the other side of the spectrum, if you eat pizza at a restaurant, you can only pick between all the restaurants offering pizza, and eat one of the pizzas they offer.

To be precise, the last two statements have to be slightly modified: At home, you can make whatever pizza you want only if you are able to, i.e. have the required knowledge, skills and ingredients. And at least in better (often more expensive) restaurants, within certain boundaries, you can actually get modifications to either the pizzas offered or the setting you consume them in, sometimes at an extra cost.

This analogy carries over fairly well to the computing world. For example, assume a company considers different alternatives to run its inventory management system. Once again, generally speaking, the level of convenience increases from the left to the right (from traditional on-premise to SaaS), while the level of control decreases from the left from the right. Or more specifically, in the on-premise approach the company can create exactly the kind of inventory management system it wants, with whatever specific features are required, and operate it with the exact level of performance, scalability, fault tolerance etc. as desired. Just like above, this is limited by the capabilities of the company - from IT skills to necessary funds. On the other side of the spectrum, if the company chooses to use SaaS, then it is limited by what the vendors of SaaS solutions for inventory management offer. And again, just like in the pizza example, for an extra fee, the company might be able (within certain limits) to get its specific needs met by the provider.

One area of the analogy is particularly tricky - cost. Generally, it seems sensible to say that making a pizza at home is cheaper than eating a pizza at a restaurant: The ingredients and the required electricity cost significantly less than a restaurant meal. However, such a comparison makes assumptions. For example, it relies on the fact that the person owns an oven. Also, it assumes that the manual labor required to make the pizza is free. This translates quite well into the IT world - if a company already owns the necessary hardware to run an application, and if operating it creates no additional labor costs (e.g. because the IT department can handle the new application without additional personnel), then running the application on-premise is likely going to be cheaper than any of the cloud models. Of course, in practice, these assumptions do not usually hold. The hardware resources used to run applications create depreciation costs over time, and operating an extra application creates extra work that costs money. Consequently, it depends on each specific situation which approach is most cost effective, with a cloud computing model likely being cheapest in many cases due to the arguments discussed in Section 1.2.2.

In closing, it should be pointed out that cost is only one of many factors that influences whether cloud computing should be chosen, and if so, which cloud service model is most appropriate in a given situation. A more comprehensive analysis of different factors will be discussed in Section 8.3.2 on cloud migration. For illustrative purposes, we will look at one more factor here, the level of control that an organization has in the different approaches. For example, irrespective of how an application is run (in-house, or in the cloud with IaaS, PaaS or SaaS), it is likely to depend on a number of platform components. For example, a web application might run on the LAMP stack (Linux, Apache, MySQL and PHP). Any of these elements will require regular updates to fix discovered security vulnerabilities. In the on-premise and IaaS models, updates to these elements are in the responsibility of the company that uses the application. In the PaaS and SaaS models, the updates are performed by the cloud provider.

A company that performs platform updates itself has more control over them. In particular, it can make sure to carefully test the updates with its applications to discover possible unintended side effects. On the flip side, it also has the responsibility to do the updates. Until tests are completed and the update performed, the application is vulnerable to security risks. So to decide between the different ways to operate the application, the company should consider carefully whether it has the resources to perform the necessary tests and updates in a timely manner. If so, this would be one argument towards on-premise or IaaS hosting, because only in these models application-specific compatibility tests of the platform updates are performed. If not, it might be better to utilize PaaS or SaaS, because the cloud service provider is likely able to perform a wide range of tests on the updates, and install the updates, in a timely manner. The tests of the provider (at least in PaaS) would not be specific to the application (which the cloud provider is not responsible for, after all), which would raise a small risk that potential issues are not discovered. However, this risk is probably preferable to the risk of either installing updates without any significant tests, or the longer exposure to the security vulnerability if the updates are delayed.

4.2 Types of Cloud Services

This section gives an overview of the range of cloud computing services that are available, using the Google Cloud Platform (GCP) as an example. Note that other cloud service providers such as AWS offer a very similar range of services (see e.g. here[2] for a comparison of AWS and Google Cloud Platform services).

The services can be categorized as follows:

- Computational Services
- Network Services
- Storage Services
- Big Data
- Machine Learning
- Management Services

When an organization uses the Google Cloud Platform, any individual resource (such as a virtual machine or a storage bucket) is associated with a single project. Projects are used to manage billing, access permissions, track use via budgets and quotas etc. Customers can simply use one or several separate projects for this purpose, or create an organizational node that can contain projects either directly and/or group them in a hierarchy of folders.

GCP resources are hosted in a specific region (e.g. us-west1). Regions contain multiple zones that are physically independent from each other (e.g. us-west1 a,b,c), which means they are typically located in separate data centers. Therefore, a failure e.g. due to a fire in one zone should not affect the other zones in a region. This allows customers to place their cloud

[2]https://cloud.google.com/docs/compare/aws

resources deliberately to increase fault tolerance and availability of their applications.

All cloud resources can be managed either through the web-based GCP console, on the command line via the GCP Cloud SDK (Software Development Kit) or the Cloud Shell, through REST-based APIs or (with limited functionality) in GCP mobile apps.

4.2.1 Computational Services

Computational services allow cloud users to run applications in the cloud. In practice, this happens at one of the following four levels of abstraction: Infrastructure as a Service, Containers, Platform as a Service or Functions as a Service.

Infrastructure as a Service (IaaS)

Computing using IaaS means the use of virtual machines in the cloud. A wide range of (virtual) hardware configurations are available, typically ranging from a single CPU core with less than 1 GByte of memory (costing less than a cent per hour) to more than a hundred CPU cores and several Terabytes of memory (costing around $20 per hour). Additionally, many different configurations with regard to storage (size, use of hard disks vs. solid state drives), GPUs etc. are possible. Virtual machines are available with a range of different (usually server) operating systems, including various Linux distributions and Windows Server. Of course, the user can also use their own operating system and configuration using pre-configured images. Virtual machines can be created very rapidly (often in less than a minute).

IaaS computing services usually offer features beyond bare virtual machines, including networking functionality such as firewalls. Another common feature would be automatic scaling. In GCP, users can define instance templates, which specify configuration details (e.g. using startup scripts) so that instances can be created automatically using the template. Using such an instance template, an instance group can be defined. Used in combination with auto-scaling load balancers (a networking feature in GCP), instance groups can be configured to e.g. automatically scale the number of web server instances serving a domain up and down based on load. Users can furthermore specify in which server center locations different virtual machine instances should be created. This allows users to make their applications extra resilient in the face of e.g. natural disasters, which typically only affect a single location. Even though IaaS-based computing is conceptually very similar to running virtual machines in an on-premise server center, these advanced features offer significant improvements in rapid scalability and reliability.

A final interesting feature is the availability of discounted virtual machine instances. The GCP offers spot instances (previously called preemptible instances). The price of these instances relative to standard instances changes over time, but is usually substantially lower, reflecting e.g. a 50-75% discount. The catch is that Google might shut these virtual machines down with minimal advance warning (a brief, orderly shutdown script can still be executed). While such instances are obviously out of the question for systems that cannot afford significant downtime, they are well suited for offline processing. For example, for scientific workloads evaluating massive amounts of data that run for days at a time, it might not matter very much if the results are available a few days earlier or later. However, the cost savings might make a huge difference.

Google offers these spot instances to increase the utilization of its computing resources. Since it cannot precisely predict how many virtual machine instances its customers will need, it has to keep a certain amount of spare capacity available so that it does not run out of resources. Rather than keep this spare capacity completely idle, Google sells some of it to customers in the form of spot instances. If a sudden spike in resource use occurs, Google can simply shut down a few spot instances to create free capacity. In the meantime these instances create at least some revenue.

Even if spot instances are not an option, users can get discounts either for "sustained use", i.e. if virtual machines are running for a significant portion of a month, or for committing to using virtual machines for a longer period of time upfront (e.g. for a whole year). While such options essentially eliminate the potential advantage of elasticity in cloud use, they can still be very appropriate for stable compute loads.

Containers

Containers (using the Docker[3] standard) are another way to deploy applications in the cloud. They are more "light-weight" than virtual machines, but still contain everything an application needs to run. In this section, we just discuss the GCP solutions to manage the containers, more details concerning what is a container can be found in Section 5.3.4.

The Google Cloud Platform uses the cloud orchestration system Kuber-

[3]https://www.docker.com/

netes[4] to run and manage applications in its Container as a Service (CaaS) model. Kubernetes manages the deployment of applications in a declarative fashion - in other words, users specify what they want to achieve, rather than how to achieve it. For example, a user can specify that an application that consists of a number of containers is deployed redundantly (for fault tolerance purposes) on say two different nodes (i.e. underlying execution hardware). If one of the nodes on which that application is deployed fails (e.g. because of a network problem), Kubernetes will automatically deploy the application on another, reachable one. Similarly, it can be configured to scale the deployment up and down based on system load, to automatically roll out (and, in the case of errors, roll back) updates to the application on all the clusters on which it is deployed etc.

The Google Kubernetes Engine (GKE)[5] frees the user from having to manually deal with deployments and the underlying resources that support them[6]. Therefore, it can be considered to be at a higher level of abstraction than using bare IaaS, where the user is responsible for allocating required virtual machines and for deploying the application on them. Another significant advantage is that vendor lock-in is not a big concern since Kubernetes is an open-source system that can easily be migrated to other underlying platforms.

Google Cloud actually offers another way to run applications in containers - Google Cloud Run[7]. In contrast to the GKE, Cloud Run offers users less influence over configuration and management of the underlying clusters, and instead manages these aspects for the user, which might make it a better choice for simple applications.

Platform as a Service (PaaS)

In the PaaS approach, the responsibility for the underlying infrastructure lies with the cloud provider, which frees the user from having to worry about it. The Google Cloud Platform's PaaS offering is the App Engine.

In the App Engine, users can deploy applications using a range of different programming languages, including Python, Java, PHP and Go. Each application consists of a number of services, which in turn can exist in several versions. Individual service versions are deployed in instances. The

[4]https://kubernetes.io/

[5]https://cloud.google.com/kubernetes-engine

[6]In fact, the Google Kubernetes Engine creates virtual machines in the Google Compute Engine to run the containers on. The advantage to using IaaS is that the user does not have to deal with these virtual machines directly at all.

[7]https://cloud.google.com/run

number of deployed instances can be configured manually, but automatic scaling based on load is also available.

The App Engine offers two different environments, the standard and the flexible environment. The standard environment is limited to specific supported programming languages in specific versions, and offers only limited configuration options. On the other hand, it is fully managed, and can scale down to zero, i.e. does not incur any costs if the application is not invoked at all. The flexible environment uses Docker containers, and is consequently not limited to a fixed list of programming languages and versions. It also offers more configuration options. However, while it can scale up and down, it incurs some costs even if idle, since at least one instance is always running, and it requires more configuration efforts from the user.

PaaS is a serverless approach. This term is technically incorrect - at the end of the day, deployed applications are still executed on servers (like any other type of application as well). The point is that in PaaS, the developers and operators of applications do not have to manage the servers themselves - this task is taken over by the cloud service provider. The App Engine is not the only serverless compute offering of the Google Cloud Platform. Cloud Functions (discussed below) are another option. The difference is that in the App Engine, the unit of deployment is usually an application that consists of multiple functions and that needs to maintain some context that survives beyond individual function calls. Typical examples for applications that are well suited for the App Engine include simple web applications or backends for mobile applications.

Functions as a Service (FaaS)
As discussed, FaaS is another serverless approach. Here, users write individual functions in languages like JavaScript (using the Node.js framework), Python, Go or Java. For each function, triggers are defined, such as http requests, specific cloud storage events, or other specific messaging or logging events. When a trigger event occurs, the corresponding cloud function is executed. Just like with PaaS, the user does not have to worry about provisioning the computing resources that execute Cloud Functions, and is only charged for the computing time that actual function invocations consume. Note how this is very different from e.g. the IaaS model, where the user allocates a number of virtual machines, and is charged for them (irrespective of whether they are busy or idle).

A typical example use of Cloud Functions could be a website or mobile app that allows end users to upload photos. Whenever a photo is uploaded, it has to be processed, e.g. optimized, smaller versions of the photo (including thumbnails) have to be created etc. Putting this functionality into a cloud a Cloud Function makes the application very scalable and cost effective. Cloud Functions can also be very useful to combine different cloud services - e.g. when changes in storage occur, processing steps can be triggered.

4.2.2 Network Services

The different types of cloud services described in this chapter are usually not used in isolation. For example, users of compute services often use storage services to store the data involved in their computations. Similarly, networking services are usually not used as a stand-alone services, but as an enabling feature that makes using just about any other cloud service type possible in the first place. To put this differently: Cloud network services are used to connect the other cloud services (such as compute or storage) with each other, with their end users and any in-house IT resources an organization might have (for example in a hybrid cloud scenario).

Virtual Private Cloud (VPC)
In the Google Cloud Platform, all cloud resources of an organization are by default placed into a Virtual Private Cloud (VPC). A VPC has global scope, i.e. can contain resources in any GCP data center in the world. VPCs offer managed networking functionality. In particular, this means that internal routing does not have to be configured manually. In fact, (virtual or physical) networking equipment such as routers or switches do not have to be provisioned or managed by cloud users at all.

By default, the network topology is managed automatically. For example, a subnet is created for each region, as subnets cannot span multiple regions. Organizations can also choose to define their network structure manually, which puts them in charge of defining appropriate subnets, IP ranges, firewall rules etc.

Cloud resources in a VPC have an internal IP address through which they communicate with other resources in the same VPC. To be accessible from the outside, they can also have an external IP address. External IP addresses can be ephemeral, i.e. assigned from a pool of addresses that are shared, and that consequently typically change if a resource such as a virtual machine is shut down and later restarted. Alternatively, for an extra charge, static IP addresses are available, which are reserved for a specific resource.

VPCs are protected by a distributed stateful firewall. Rather than having to manually create and manage virtual or physical firewalls between different cloud resources and other networks, firewall rules apply to the network as a whole, and allow or deny network traffic at the instance level. By default, all incoming data traffic (ingress) is denied, while all outgoing traffic (egress) is allowed. More specific firewall rules can be defined by users, specifying whether traffic is allowed or not based on its source or destination, protocol and port used. Firewall rules have a numeric priority attached to them, and are applied in priority order. The first applicable rule decides whether traffic is allowed or denied.

As mentioned above, resources in GCP are grouped into projects. VPCs can be shared by multiple projects in the same organization. Alternatively, multiple VPCs (containing projects from the same or different organizations) can use network peering to connect to each other, which allows for decentralized network administration.

Load Balancing

Load balancing (combined with the ability to scale computing resources up and down easily) is one of the core functionalities required in cloud computing. Load balancers enable the operation of applications with loads that exceed what a single server (even a powerful one) can handle. Such applications are very common - for example, any busy online store or other web site probably needs multiple web servers to handle incoming requests at peak loads. A load balancer distributes all incoming requests between whatever number of servers is necessary to handle the load.

The use of load balancers not only helps ensure adequate performance (e.g. low response time) of applications, it also increases their availability: Load balancers can detect malfunctioning servers, and forward incoming requests to any of the other servers. Of course, a poorly designed load balancer itself can become a “single point of failure”. In other words, if the load balancer is implemented as an application on a single machine, a failure of that machine makes the whole application unreachable for all users, even if the actual servers behind the load balancer all work properly. Needless to say, the load balancers offered by GCP do not suffer from this problem, but are instead built with appropriate redundancy.

GCP offers different kinds of load balancers for different kinds of purposes. There are load balancers for HTTP(S), SSL and TCP traffic that are global in scale, meaning they can forward requests to servers in different regions based on proximity to the user. There are also network load balancers that work on a regional level and can handle any TCP or UDP traffic, both for internal only and for external traffic.

Other Networking Features

GCP offers several other types of networking features. Cloud Interconnect allows organizations to connect their internal (on premise) networks to GCP VPC networks, either directly or through interconnect partners.

Especially for organizations that have users in many different locations, Cloud CDN (Content Delivery Network) allows to cache content closer to users, reducing end user latency and network traffic cost.

Cloud DNS is a scalable Domain Name Server that organizations can use to resolve domain names to IP addresses.

Finally, GCP offers a number of networking security features, including Cloud Armor, which helps mitigate potential DDoS (Distributed Denial of Service) attacks, as well as other common types of network based attacks such as cross-site scripting (XSS) and SQL injection attacks.

4.2.3 Storage Services

Just like networking, most cloud applications need storage services as well. Traditionally, storage meant file storage, and later storage in relational databases for structured data in table format. Today, a much more diverse range of storage options exists in the storage field in general, and this is reflected in storage cloud services as well.

Cloud Storage

GCP Cloud Storage allows organizations to store unstructured data in objects. This is used by Google itself for many of its other services - for example, Google Photos stores user photos and videos and Gmail stores email attachments using Cloud Storage.

Cloud Storage organizes data in buckets, which have a unique name, a location (in one or multiple regions), a storage class, access policies and object versioning and lifecycle management options.

There are four different storage classes:

- Standard Storage
- Nearline Storage
- Coldline Storage

- Archive Storage

Standard Storage is the default storage class for storing data that is accessed regularly. While it costs the most for storage (per GB per month), it features the lowest access costs. It also has no minimum storage duration. Nearline, Coldline and Archive Storage offer increasingly cheaper storage rates, but charge increasingly more for data accesses. They also feature minimum storage durations of 30, 90 and 365 days, respectively, which means that while storage buckets and their content in these classes can be deleted at any time, the user is charged the minimum number of days as indicated.

All storage classes can be used within a single region, or in two or multiple regions. The latter offer slightly more reliability due to the extra redundancy, but also cost slightly more.

Relational Databases
Relational database systems are still the backbone of many business systems that deal with structured data. Therefore, it is not surprising that GCP offers relational databases in the cloud.

Cloud SQL offers users fully managed MySQL, PostgreSQL or SQL Server instances that handle replication, backups and automatic scaling for the user, supporting up to 64 CPU cores, 400 GByte of RAM, and 10TB of storage.

Users that need larger instances can instead use the Cloud Spanner service that, thanks to horizontal scalability, can be used for much larger applications.

Non-Relational Databases
Cloud Datastore is a NoSQL database. It doesn't require users to define a database schema, supports Atomic, Consistent, Isolated and Durable (ACID) transactions, and is fully managed, i.e. automatically taking care of scaling, replication etc. It supports SQL-like queries.

Compared to Cloud Datastore, Cloud Bigtable is meant for even larger amounts of data. It is another NoSQL database offering for large workloads. It can scale to billions of rows and thousands of columns of data. It is also fully managed, and integrated with open source big data tools like Hadoop.

Data Warehousing
BigQuery is another offering for very large amounts of data. In contrast

to the non-relational databases from the previous section, the focus of BigQuery is on Online Analytical Processing (OLAP) type queries, rather than on Online Transactional Processing (OLTP) queries. In other words, it performs best if used in queries that summarize large amounts of data (e.g., how do the sales of products in category X in Q1 of 2019 compare to Q1 of 2020?), rather than on large numbers of individual queries (e.g. inserting thousands of new sales transactions per hour).

Block Storage
Block storage is a technique where any data (like a file or database entry) is divided into blocks of equal sizes. Then, the block storage service decides where these blocks must be distributed in the cloud in order to optimize the access time. In particular, Block storage service is useful when application is latency sensitive and it requires high-performance. Comparing with file storage, Block storage architecture provides multiple paths to the data and this make this solution more flexible and scalable with respect to file storage where only one path is provided to retrieve data.

Solid-State Drives Vs Hard Disk Drives
In some of the storage services discussed above, the user has to select between Solid-State Drivers (SSD) or Hard Disk Driver (HDD) when they provision a storage service (like Relational Databases). The option to prefer depends on how different factors: amount of data to store, data access frequency and performance requirements . In general, SSD storage is the most efficient and cost effective choice, while HDD storage is appropriate for large dataset (>10TB) where the latency is not important and the data requests frequency is low.

4.2.4 Big Data

With the three types of resources discussed above - compute, networking and storage - any kind of application can be built, and users take advantage of that to build and operate a wide variety of applications in the cloud. It has become apparent that cloud computing is often particularly beneficial for specific kinds of applications, and, consequently, modern cloud providers offer specialized features to support them.

One of these can be largely summarized as Big Data - or by related terms like Business Analytics, Data Mining, Business Intelligence, or whatever the most fashionable buzzword is right now. The essence of these kinds of applications is that they feature massive amounts of data - often in the range of Petabytes - that are to be analyzed to gain valuable insights from.

The idea behind Big Data is that most organizations have massive amounts of data from their normal operations. Consider an online retailer, who would not only have all the sales transactions, but probably also data about the click streams of visitors of their web site, whether the visit resulted in a sale of not, and of course a wide range of other relevant data that can easily be acquired, such as economic or even weather data (factors that would also influence online sales). Valuable insights for such an online retailer might be predictions of what kinds of products might be popular in the next season (so that enough inventory can be acquired), how to price items to maximize sales or profit or whatever else the goal is, etc.

One of the challenges of Big Data applicants is that they require very significant IT resources, not only with regard to storage, but also with regard to parallel processing and consequently networking, in order to be able to run analyses on massive amounts of data in a short period of time (such as minutes or hours, rather than days or weeks). Yet, in many cases, an organization would likely not be using these significant (and expensive) IT resources every day of the year, all day long, but probably only for a fraction of the time, when executives want analytical questions answered. This potential under utilization of expensive IT resources is an important argument why operating Big Data applications with a cloud service provider rather than on-premise is likely very cost effective.

The Google Cloud Platform offers three different services that support Big Data applications: BigQuery, Cloud Dataproc and Cloud Dataflow.

BigQuery

The observant reader might notice that BigQuery has already been discussed at the end of the last section about storage options in GCP. This is no accident - it fits in both areas. BigQuery can primarily be used to store large amounts of data, in which case it belongs in the storage category. On the other hand, if its query features are used extensively, it falls more into the category of Big Data.

As discussed above, BigQuery is a modern data warehouse solution, which means it is not so much meant to run transactions that support day-to-day business operations, but to perform larger analytical queries on relatively stable data. BigQuery is a fully managed service, meaning that storage management, backups, software updates, query optimization etc. are done by the platform. The user also does not have to provision resources such as storage capacity and clusters of virtual machines to process queries etc. Instead, resources are automatically provisioned as needed, and the user is only charged for the resources that are actually consumed.

Cloud Dataproc
Cloud Dataproc allows users to process large amounts of data with fully managed compute clusters using e.g. Apache Hadoop or Spark in the cloud. The basic idea of these tools is the map-reduce paradigm - to parallelize large computing tasks on massive amounts of data by breaking them into many smaller jobs that are processed in parallel, and to then collect the results and to combine them to the overall result in the end. The obvious advantage of this approach is that the processing is significantly sped up compared to serial processing on any individual machine, no matter how powerful that machine is.

This general approach can be used for many different applications, from complex queries over batch processing to machine learning. Once again, cloud computing lends itself very well to this paradigm, because running e.g. fifty virtual machines for an hour costs the same amount as running a single virtual machine for fifty hours. In other words, a task that can be parallelized well can be sped up massively compared to serial processing, at no significant increase in cost. Additionally, since users do not have to pay for idle resources, significant cost savings compared to operating similar clusters in-house can usually be realized, especially if tasks are not time critical and can be interrupted, in which case even cheaper spot instances can be used.

Cloud Dataflow
Cloud Dataflow is meant for processing streams of data in real time. This can be used in many different contexts, from ETL (extract, transform and load) tasks that move data between different data stores over batch processing of large amounts of data to processing of data that naturally arrives constantly over time such as sensor data. In Cloud Dataflow, users can define pipelines of operations on data that are processed in real time. These operations can include map/reduce steps. One of the big differences to Cloud Dataproc is that Cloud Dataflow is serverless. In other words, users do not have to allocate resources for their processing steps. Instead, the resources used for individual pipeline steps scale up and down automatically based on current demand.

4.2.5 Machine Learning

Machine Learning is an Artificial Intelligence (AI) approach that uses neural networks (which mirror, in a simplified way, how the human brain works) to build models using training data to make "intelligent" predictions or decisions - e.g. about whether an email is spam, how to transcribe a spoken sentence into written language, or whether a person in a photo is smiling. These systems do not use explicit programming, which, in the third example, might look for say roughly u-shaped lips. Instead, the neural networks are fed with a large number of sample pictures of people. All the photos in this training data are sorted by human work into two groups - smiling or not smiling. The machine learning algorithms modify the detailed configuration of the neural network until it predicts with high accuracy whether a face in fact shows a smile or not.

Machine Learning requires large amounts of training data (often tens or hundreds of thousands of samples), and they require significant computing resources to train the models with the training data, and, to a lesser extent, to apply the resulting model to the actual data to be analyzed. These requirements make them excellent candidates for cloud computing.

The Google Cloud Platform features a range of Machine Learning offerings under the umbrella of Vertex AI [8]. This is an area with rapid technological progress, so new features and offerings are released rapidly and often. The following are some of the main features:

- Vertex AI Model Garden allows users to find and try out a wide range existing machine learning models, to either use directly or adapt for a specific task.
- AutoML helps users with limited machine learning expertise to develop their own machine learning models
- VertexAI Workbench is an integrated development environment that supports users with their whole data science workflow
- Numerous tools like Vertex AI Pipelines and Vertex AI Model Monitoring support users with the whole machine learning life cycle, for example by enabling the automatic re-training of machine learning models when their precision declines with the availability of new data.

[8]https://cloud.google.com/vertex-ai

Two different types of machine learning models are generative AI models and predictive AI models. Generative AI can be used to create content, e.g. to auto-complete code, chat with a user, answer questions, create images based on user prompts, create short summaries of longer texts etc. Predictive AI models are used in a wide range of applications, such as predicting if a credit card transaction is likely fraudulent or not or whether a photograph of a persons skin displays signs of skin cancer or not.

Pre-Trained ML Models In some cases, users need a Machine Learning solution for which GCP already offers a ready-to-use pre-trained models that can be reached via ready-to-use application programming interfaces (APIs). These models include:

- The Vision API allows users to do standard tasks on images, such as detecting faces, judging facial expression, detecting famous landmarks, extracting text etc.
- The Speech-toText API converts speech to text and vice versa.
- The Cloud Translation API translates text between different languages.
- The Cloud Natural Language API helps users analyse e.g. the syntax, content or sentiment of unstructured text.
- The Video Intelligence API can be used to detect and label the content of videos.

4.2.6 Management Services

This final section describes cloud services that are not meant to be useful on their own, but to help manage the use of cloud services such as compute, storage, networking etc. The Cloud Deployment Manager supports users with managing their cloud infrastructure. Stackdriver is a tool that allows users to monitor their applications, including more advanced features that can help with analyzing and debugging errors, identifying and solving performance issues etc.

Cloud Deployment Manager Infrastructure as Code (IaC) refers to an approach that treats the structure of IT infrastructure like other software code. Only in the cloud is this fully possible, where resources are generally virtual, which allows them to be fully software defined. For example, an application might be designed to run on a dual-core machine with 8 GBytes of main memory, a 500 GByte solid state drive and a specific Linux version installed. If this application is operated on a physical machine on-premise, then only the details of the software installation can be defined in code (e.g. startup and shell scripts), while the physical hardware by necessity has to be installed "by hand" - including connecting the machine to a network etc. If the same application is run in the cloud, it could be deployed on a virtual machine. While the creation and configuration of the virtual machine can be done "by hand" through the GCP console (a web-based user interface), it is also possible to define it via code. This way, the whole application, including the virtual machine type, its software configuration, its network connectivity etc. can be defined in code.

Defining the infrastructure in code has significant advantages, especially if the code is treated like any other important source code and managed in a source control system with versioning etc. For example, it ensures consistency and repeatability and reduces the amount of work necessary when infrastructure needs to be created. A typical operational system would not only be needed in a production environment, but also in separate (but ideally identical) environments for bug fixing purposes or for the development of new features. Creating these additional environments manually not only requires significant manual work for each new environment, it also increases the risk that errors or omissions happen, which would make the environments slightly different, resulting in possible problems when bug fixes or new features are deployed in production.

The GCP Cloud Deployment Manager allows users to manage infrastructure in the Google Cloud Platform via code. It uses an easy to read and write YAML (YAML Ain't Markup Language) based declarative language. In contrast to imperative languages which specify exactly how to achieve an outcome, a declarative language specifies the desired outcome, such as a virtual machine of a specific machine type in a specific region and connected to a specific network. Based on such a declarative configuration, the Cloud Deployment Manager figures out and executes the specific instructions required to create the desired state.

Google Cloud Operations Suite The GCP Operations Suite (formerly known as Stackdriver) is an integrated monitoring solution for the Google Cloud Platform. Monitoring is an extremely important aspect of operating any productive IT system. Its purpose is generally to alert operators of any issues with an application, ranging from suspicious activities that might be a sign of a security problem over slow system response times to system failures resulting in a completely unavailable application. Rather than being alerted of any such issue by angry end users who eventually email somebody about problems they face, monitoring should ideally catch problems before they are noticeable to end users.

The GCP Operations Suite offers several different components with a wide range of features:

- **Monitoring** is used to track aspects such as whether systems are up or down, how high the load of systems and their components is and what kind of user response times applications have. Alerts can be defined that trigger if specific situations occur (e.g. if a server is not reachable or if the CPU utilization of a machine exceeds a certain threshold for a specified amount of time). These alerts can not only be displayed on administrative consoles, they can also be sent via email, text message etc. to alert on-call system administrators. Monitoring not only displays current values, it also keeps track of how values change over time, which is valuable information when potential issues are investigated.

- **Logging** integrates logs from all GCP resources. Needless to say, large-scale cloud deployments can generate massive amounts of log entries. Consequently, the logging component has to be and is highly scalable.

- **Error Reporting** provides instant notifications of application errors. While the number of errors should be significantly lower than the number of log entries, large scale applications can still experience significant numbers of errors, so the error reporting component allows real-time aggregation of errors and other features that allow users to get an overview of all occurring errors, and to prioritize the most urgent or important ones.

- **Tracing** helps with the identification of performance bottlenecks. It can be used to break down the response latency of individual requests and to aggregate the information of many requests.

- The **Debugger** allows developers to debug the production system in real time. For example, snapshots of running applications showing the call stack, the contents of variables etc. can be taken. Needless to say, this can be very helpful for identifying and solving problems.

- The **Profiler** lets developers observe the performance of code in the production environment under production conditions. While performance tests that are performed before a system goes into production should give developers significant insights into how their application will behave in production, it is very hard to predict and simulate the exact request distribution, system load etc. that the system faces in real world use in a test environment. Therefore, insights gained from the profiler can be very valuable to improve code performance. Of course, collecting the necessary data for this in a running system creates some overhead. The Profiler minimizes this impact by very efficient instrumentation and through the use of statistical sampling.

4.3 Summary

Even though this chapter could only provide an overview, it hopefully successfully made the point that modern cloud computing offers a much wider range of cloud services than just virtual machines and file storage. These services include advanced networking features enabling enhanced scalability and security of applications, a range of flexible and high performance storage solutions and various other advanced services in areas such as Big Data and Machine Learning. While users can (at least theoretically) implement any of these services on their own on cloud-based virtual machines, using the available more advanced solutions is in most cases more efficient and effective. An analogy from the programming world would be that any application can be built with nothing but assembly language code, but that using higher-level programming languages and existing frameworks, libraries and tools usually saves a lot of time and effort and leads to a better outcome.

Chapter 5

Cloud Architecture

This chapter is about how to build applications in the cloud. It cannot cover this topic exhaustively. Not only is there a wide range of aspects involved, such as analysis, design, implementation, testing and operations, but there are also several different architectural approaches that can be used, each of which could easily fill not just a chapter, but a whole book. Therefore, the goal of this chapter is to give an overview of some of the most important architectural approaches, and to provide some guidance on their relative strengths and weaknesses. Additionally, several aspects that apply across these different approaches are covered.

5.1 Architecture Goals

Before specific architectural principles and approaches are discussed, this section briefly covers some of the most important goals that architectural decisions strive to achieve. While many of these goals are applicable for applications that do not run in the cloud as well, we analyze them specifically with a focus on cloud computing. The discussion will illustrate that there are several potential conflicts between the goals, meaning that it is difficult to attain all of them simultaneously.

5.1.1 Performance

One of the most obvious goals of any application architecture is that it ensures adequate application performance. The vast majority of applications today is of an interactive nature (as opposed to being batch based). In such applications, a key aspect of performance is low response times. Additionally, performance under load is important, which means that the response times for individual users should remain consistent and low even if the number of concurrent users increases.

Performance is particularly important for many cloud-based applications, because they often have global reach and rather volatile usage rates. Thankfully, cloud providers offer many features that help ensure smooth performance, from the availability of multiple data centers across the globe (and consequently close to many customers) to performance enhancing features like content delivery networks and systems that automatically scale up and down with demand.

5.1.2 Reliability

Another important goal is reliability - that the system is available and working for its users at all times, or at least almost at all times. To achieve reliability, a system needs to be fault tolerant. Even in the cloud, individual components such as servers, networking components, or even software modules can and do occasionally fail. A good cloud architecture takes this into account and designs applications in such a way that the failure of individual components does not affect the overall availability of the system.

A good analogy for the desired behavior is the contrast between pets and cattle, as explained in a discussion of the cloud architecture used by Netflix[1]: The owners of pets know and care about their individual pets. If a pet dies, its owners are sad. A farmer who has a hundred cattle does not really know or particularly care about individual animals. If a cow dies, the milk production might go down by 1%, but that does not affect the overall output significantly, and can easily be rectified by buying another animal. To achieve a reliable system, individual servers must be like cattle, not pets, in other words, easy to replace. In order to ensure that this paradigm is strictly followed, Netflix uses the concept of the Chaos Monkey[2] - a system that randomly and unpredictably shuts down individual servers. A reliable cloud architecture should not have any individual components the failure of which would result in system outages, and should automatically replace any failing components.

[1]Tilkov, S. "The Modern Cloud-Based Platform", IEEE Software, March/April 2015

[2]More details here: `https://netflix.github.io/chaosmonkey/`

5.1.3 Low Administrative Overhead

With the help of modern software development frameworks and libraries, applications can often be developed with surprisingly small amounts of code. In contrast, operating the infrastructure to run a modern web-based application can take significant amounts of effort. A technical report by the University of California at Berkeley[3] provides a good example for this: Imagine the backend for a smartphone application that wants to automatically create thumbnails of photos uploaded through the app and place them on a website. While this functionality could probably be implemented in a few dozen lines of code, creating and managing an appropriate server infrastructure for such an application would take a lot more effort. For example, the server infrastructure would have to contain sufficient redundancy to ensure that individual failures could not take down the service. Appropriate measures would have to be taken so that the backend scales up and down with the load of user requests, appropriate monitoring and logging would have to take place to ensure the availability and security of the service, maintenance tasks such as security patches of all involved components would have to be performed regularly etc.

This example illustrates that a good architecture should minimize the necessary administrative work when possible. One approach to achieve this is the so-called Serverless Computing. The term is slightly misleading - no matter what architecture is chosen, computing tasks are ultimately executed on servers. Serverless Computing refers to the approach that the user of a cloud service (which in the example above would be the developer of the smartphone application) can run the server side of their application on a cloud service that automatically manages the underlying servers and all their maintenance tasks without user interaction. In other words, if you use Serverless Computing, you do not have to worry about the servers that do your work - somebody else makes sure they work, receive security patches, are scaled up and down with the current load etc.

5.1.4 High Level of Control

Another important consideration of organizations using cloud computing services is the level of control that they retain over their cloud resources and applications. Sometimes this is driven by regulatory requirements. For example, organizations in sectors such as health care might be required

[3]Jonas, E. et al. "Cloud Computing Simplified: A Berkeley View on Serverless Computing", Technical Report No. UCB/EECS-2019-3, February 10, 2019

by law to have full control over where their data is stored and how it is protected from unauthorized access by third parties. Specific regulations might prevent the use of storage services in countries that have lower privacy standards, or might dictate the use of certain encryption standards such as specific encryption algorithms and key lengths.

Another important consideration is that of vendor lock-in. Vendor lock-in describes the situation that users of a service might find it prohibitively difficult or expensive to switch to a different service provider. Since the use of public cloud services usually does not entail long-running contracts, it might seem that vendor lock-in might not be a big concern. However, the fact that most of the services of the major public cloud service providers are not standardized often makes the proposition to switch an application from one provider to another a very difficult one. The more control cloud users have over the services they run in a public cloud, the easier it should be to switch to a different provider. More details concerning vendor lock-in are discussed in Section 1.3.2, and tools able to reduce the problem are discussed in Chapter 7.

5.1.5 Support Modern Software Engineering Approaches

The architecture of cloud applications should also be compatible with modern software engineering approaches. For example, the incremental development often practiced in agile development approaches should be supported. This ideally includes the independent development and deployment of different parts of the application, along with automatic test suites etc.

Another important current software engineering paradigm is DevOps[4], which attempts to better integrate development and operations of applications in order to increase agility, efficiency and quality. This requires architectures that allow applications and changes to them to easily be moved from development to test and production, and that already consider operational requirements and efficiency at the development stage, rather than treating them as an afterthought.

5.1.6 Cost Effectiveness

A final important goal is to have an architecture that allows for a cost-effective operation of the application. Since costs in cloud computing are

[4] Zhu, L. et al. “DevOps and Its Practices”, IEEE Software, May/June 2016

directly related to the incremental use of computing resources, this means that the architecture should use resources efficiently. In the face of often highly variable application utilization over time, this means that the architecture should allow for smooth and efficient scaling up and down of resources based on demand. In addition to that, it should make resource use and consequently costs easy to monitor, manage and ideally even to predict.

5.1.7 Goal Conflicts

As indicated above, there are some clear conflicts between these goals. For example, using as few computing resources as possible improves cost effectiveness, while it tends to run counter to performance and to high reliability through redundancy. Similarly, low administrative overhead can be achieved by building an application on top of cloud services that are mostly managed by the cloud provider. However, such an approach reduces the amount of control that the cloud user has over the underlying infrastructure.

Since all the goals cannot fully be achieved at the same time, a good cloud architecture should strive to find a good balance between the goals, taking into account the specific priorities of the application under development.

5.2 Architectural Principles

This section describes several architectural principles that many modern cloud applications use. They are about different aspects of cloud architecture, which means that they are not mutually exclusive choices. Instead, several principles can be combined in different ways to achieve the goals of a specific application in a given situation.

5.2.1 Infrastructure as Code

In traditional, non-cloud applications, there is a pretty strict boundary between software and the underlying physical infrastructure. Software includes a wide range of elements from source code, configuration settings, application specific data (e.g. product details, prices etc.), website layouts, images etc. The common aspect of all these elements is that they can be stored in files or version control systems, that they can be backed up and restored, or recreated in a different environment without any physical access or changes to physical hardware such as servers and networking equipment. The physical infrastructure is very different. For example, if an organization wants to clone the production environment into a separate environment, for example to help with debugging and bug fixing activities, this requires someone to set up an additional environment by physically installing a new set of servers, connecting them (mirroring the setup of the production environment) through network equipment such as switches and routers etc.

One of the key benefits of cloud computing is that the underlying infrastructure of cloud applications is typically[5] virtualized. In other words, rather than running directly on a set of physical machines, connected through a physical networking setup with switches, routers and network cables, cloud applications run in a virtualized environment, such as a set of virtual machines connected through a virtual network infrastructure. While these virtual resources of course ultimately still run on actual server and networking hardware, cloud computing users are not concerned with these physical resources. In other words, a cloud user can clone the environment in which an application runs in the cloud without touching any physical hardware (outside of the mouse and/or keyboard with which the cloning is requested).

In order to maximize the benefit of this virtualization, cloud users should treat their underlying infrastructure as code rather than just manually putting it together through the web interface of the cloud provider. This way, the infrastructure can be treated just like the software of an application - managed in a version control and configuration system, subjected to unit tests etc. This has multiple benefits such as the ability to rapidly set up or clone identical environments, to do so with minimal effort (e.g. by just running a script) and with perfect consistency of configurations.

In the Google Cloud Platform, the Cloud Deployment Manager[6] can be used to specify and create cloud resources using a simple YAML[7]-based declarative language. For example, the code to create a simple virtual machine using the machine type f1-micro with a solid state disk running the web server nginx on Ubuntu 18.04 and connected to a default virtual net-

[5]While most cloud applications run on virtual machines, either directly or through other cloud services that ultimately run on virtual machines, in specific niche cases it can make sense to run applications directly on physical hardware that is reserved for a specific customer. Consequently, cloud providers typically also offer what is called "bare metal" servers, e.g. https://cloud.google.com/bare-metal.

[6]https://cloud.google.com/deployment-manager

[7]YAML is a recursive acronym for "YAML Ain't Markup Language"

work could look roughly as follows:

```
resources:
  - name: test-instance
    type: instance.py
    properties:
      zone: us-central1-a
      diskImage: projects/ubuntu-os-cloud/images/family/ubuntu-1804
      machineType: f1-micro
      diskType: pd-ssd
      networks:
        - network: default
          accessConfigs:
            - type: ONE_TO_ONE_NAT
      metadata:
        items:
          - key: startup-script
           value: sudo apt-get update && sudo apt-get install -y nginx
      tags:
        items:
          - your-tag
```

The code of the virtual infrastructure for a real cloud based application is obviously significantly larger, and includes many other resources such as database servers and load balancers, as well as more complex settings like several independent subnets with specific firewall rules. However, with the help of modern software tools and processes, such code can be handled easily and reliably.

5.2.2 Automation

Setting up the virtual infrastructure of an application using code as described in the previous section is only one of many different aspects that can be automated in a cloud setting. In order to understand the appeal and importance of automation, it is useful to look at an example. Imagine the online store of a retailer during the peak shopping season before Christmas. As customer traffic increases to multiple times the normal load, the infrastructure behind the web store might easily scale to dozens or more virtual machines running the necessary web and application servers. If for instance a security patch for a bug in the web server software becomes available, it should (after appropriate testing) be quickly installed on all the server instances. If this had to happen manually, it would not only take a fair amount of time to complete the task on all the servers, which would potentially leave the yet unpatched servers open to attacks, it would also take up a significant amount of work by the system administration team.

Automation tools can help in such situations. Ideally, an administrator should just need to specify which kind of change needs to happen on a set of machines, and the automation tool then performs these changes. In such a setting, the human effort required to implement a change would be fairly independent of the number of affected machines. To put this differently - in an automated environment, if an application running on a specific number of servers can be managed by a team of system administrators, the same team should still be able manage the application even if it scales up to a multiple of the original number of servers.

There are numerous automation tools, for example Puppet[8], Chef[9] and SaltStack[10]. We briefly look at yet another tool, Ansible[11], as an example. The choice of Ansible should not be interpreted as an endorsement - it was chosen because it is conceptually quite simple, and because it supports the Google Cloud Platform well. However, other automation tools can be similarly effective.

Ansible (the architecture of which is illustrated in Figure 5.1, source: Dakuna website [12]) is operated through the Ansible Management Node, a computer on which Ansible is installed, and through which system administrators manage the whole automation process. On this management node, the Host Inventory and the Playbook are kept.

The Host Inventory is ultimately just a list of the machines (physical or virtual) that are managed by Ansible. In its simplest form, the Host Inventory could simply contain a list of DNS names or IP addresses. In practice, some additional features are typically used, such as organizing hosts into groups that make sense for the application (such as a group “web servers” and a group “database servers”), or storing additional information like connection types and user names to use with different hosts.

Ansible Playbooks contain different configurations that system administrators want to apply to systems. Playbooks are written in a YAML based syntax, and describe in a declarative way a desired state of a resource. In other words, they do not directly contain code or commands to be exe-

[8]https://puppet.com/
[9]https://www.chef.io/products/chef-infra
[10]https://www.saltstack.com/
[11]https://www.ansible.com/
[12]`https://danuka-praneeth.medium.com`

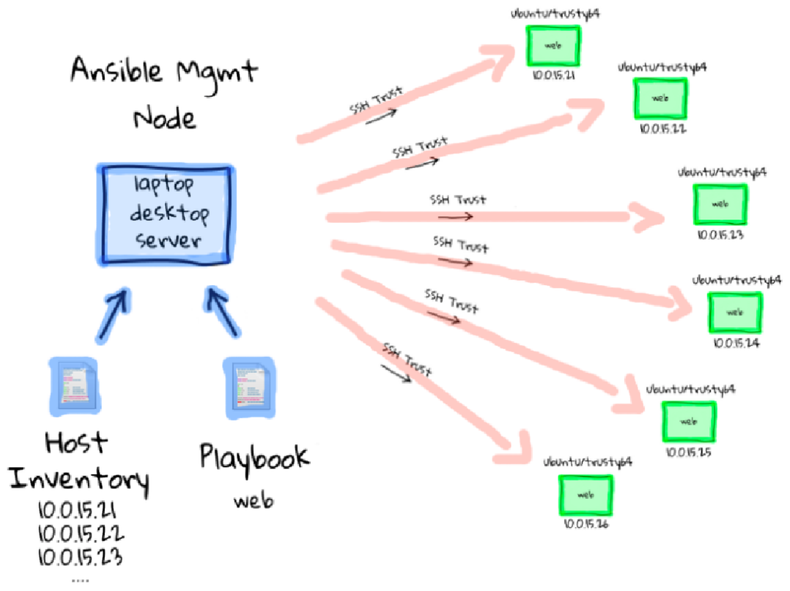

Figure 5.1: The Ansible architecture

cuted, but allow system administrators to specify which result they would like to achieve on a managed system. For example, a playbook to ensure that Apache is updated to its latest version could look like the following:

```
tasks:
  - name: ensure apache is at the latest version
    yum:
      name: httpd
      state: latest
```

A system administrator could now simply apply this playbook on the group of machines that play the role of web servers. This would cause Ansible to connect to all the machines in that group via SSH, to check the installed version of Apache, and to update it if necessary (or do nothing if it already is the latest version). The fact that Ansible uses SSH to connect to target machines and to execute whatever actions are necessary means that Ansible does not require any special management software to be installed on the target machines, only a user account with sufficient permissions to connect to.

It should be clear that with a tool like Ansible, the effort required to manage servers does not scale linearly with their number. While increasing numbers of hosts might require a little extra effort in maintaining the host inventory, the work required to maintain and apply playbooks is ultimately the same whether ten or one hundred servers are administered.

Finally, Ansible directly supports several cloud environments such as the Google Cloud Platform by providing numerous plugins that can be used to include common tasks in the cloud in custom playbooks without any development effort. For example, in version 2.10, Ansible directly supports over 170 different GCP modules, ranging from the creation of GCP App Engine firewall rules to gathering information on a GCP TPU (Tensor Processing Unit) Node[13].

5.2.3 Microservice Architectures

Another increasingly common architectural principle is to design applications as a collection of so-called microservices[14], rather than in a monolithic fashion. Traditionally, applications are built in a monolithic way, i.e. they often consist of a single executable file. While such applications might internally consist of multiple fairly independent components, for instance to facilitate the development by several development teams, there is no mechanism to independently use only a subset of these components. The application is either deployed completely or not at all, and there is no simple way to replace parts of the application by other components.

Such a monolithic architecture has significant disadvantages. For example, if any part of the application needs to be changed, the entire application has to be recompiled and redeployed. While this is not a big concern for applications that are stable and rarely changed, it is not a very good fit for many cloud-based applications, which are often developed in a very agile fashion. These are often used in an environment in which frequent small changes - bug fixes or new features - are desirable, responding to business settings in which time-to-market is often of critical importance. Another problem of monolithic applications is that they often grow over time, and any initial separation into internal components deteriorates over time, which makes the applications harder and harder to maintain.

Microservice architectures do not suffer from many of these issues. In such architectures, applications are assembled from a number of independent components (microservices), each of which is only responsible for a

[13]List of Ansible modules for GCP: `https://docs.ansible.com/ansible/latest/collections/google/cloud/index.html`

[14]`https://martinfowler.com/articles/microservices.html`

small part of the overall functionality. This makes it possible to update or even replace individual microservices without impacting the entire system. As long as the interface of the microservice stays the same, the rest of the application should not be affected and does not have to be redeployed. This allows for agile development approaches in which different parts of the overall system can be developed and maintained by different teams of developers, and in which frequent changes to the application can be done with little overhead. The boundaries of responsibilities between the different components of the overall system are also explicit, which prevents their slow deterioration like in monolithic applications.

Despite their indisputable advantages, microservice architectures also introduce new challenges. To begin with, microservice applications have to be designed very carefully. In particular, the responsibilities of individual services and their interactions have to be well thought out, or the entire application becomes a complicated mess with countless unpredictable interdependencies between the different microservices. But even if the overall architecture is well designed, challenges remain. For example, microservices have to be able to locate instances of other microservices they communicate with. This communication should be secure, which might involve encryption. Microservices also need to be able to deal with the failure of other microservices - be it because the communication fails, or because actual system outages occur. Also, different microservices might have to deal with significantly different loads in service calls, often requiring scalable deployments with load balancing. Many of these issues are made even harder by the fact that especially in larger applications, microservices might well be deployed on numerous different servers, which makes unexpected and independent failures of parts of the application much more likely.

One approach to make many of these issues more manageable is to use a service mesh such as Istio[15]. The idea behind service meshes is to introduce an additional infrastructure layer that deals with aspects such as service discovery, message routing, security aspects such as encryption, load balancing, failure recovery and overall monitoring.

Figure 5.2 (source: The Istio's website[16]) gives an overview of what a typical service mesh architecture looks like. Each service is connected to a proxy that takes care of the network traffic between different services. The services themselves and the proxies are both in the data plane in which the communication of the actual application happens. The separate con-

[15]https://istio.io/latest/docs/concepts/what-is-istio/

[16]`https://istio.io/latest/docs/concepts/what-is-istio/`

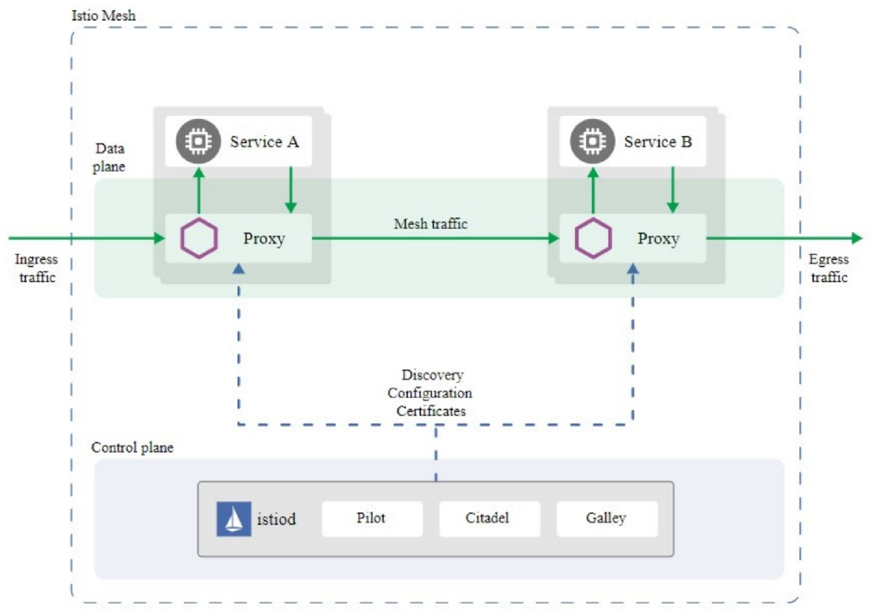

Figure 5.2: Istio Service Mesh Architecture

trol plane controls the proxies and implements higher-level infrastructure needs such failover support, load balancing, monitoring etc. The benefit of such an approach is that the actual service implementations do not have to deal with any of these issues, and instead can focus on the business logic of the application.

5.2.4 Hybrid Cloud Architecture

As discussed in the first chapter of this book, the term hybrid cloud describes an approach in which an organization uses both a private cloud infrastructure and one or more public cloud providers. Reasons for doing this include the need to utilize existing hardware while still being able to rapidly scale resources beyond the available in-house capacity in the event of sudden surges of demand. Also, relying exclusively on a single public cloud provider puts an organization in significant dependency of that provider. For example, contract disputes could put such an organization at the brink of a complete interruption of their applications, which could easily put it out of business in a matter of days.

While organizations might well choose to operate their most critical applications in-house and to put everything else into public cloud, in the context of this chapter on cloud architecture we are particularly interested in approaches where a single application is distributed across a private and at least one public cloud. The most obvious approach to do this is to deploy the whole application in the private cloud, for example on a number of virtual machines. If the load requires more capacity than can be provided by the private cloud, the application is scaled into additional virtual machines with a public cloud provider.

Another approach is to use a layered hybrid cloud architecture, for example as used by the electronics retailer BestBuy for its online store[17]. In this approach, parts of the application run only in the private cloud, while other parts run only in a public cloud. In the example of BestBuy, the frontend of their online store, i.e. the actual website with the catalogue of all the products and their details is run in a public cloud, while the backend, in particular the actual processing of customer purchases, is handled in their own datacenter.

The justification for this approach is twofold. Firstly, the frontend of the application is subject to significantly more load. Most visitors to the BestBuy website browse the products offered without actually ending up making purchases. This browsing can create significant load on the system, as customers use different ways of browsing and searching for products, invoking product comparisons etc. Given that customer demand is rather "peaky", i.e. volatile - for instance, busy in the Christmas shopping season, but not very busy at many other times, it makes a lot of sense to run this part of the application in a public cloud, where resources can be scaled up and down rapidly, and only actually consumed resources are charged. While the backend is of course also affected by the volatile nature of customer demand, the overall computational complexity of processing customer purchases once customers decide on the products they want to buy is relatively low. This means that a relatively small number of servers can handle the load, and the cost of keeping some spare capacity on these servers is limited.

Secondly, the frontend of the application is also much less critical to the business. For example, if an attacker managed to successfully break into the frontend servers in the public cloud, the main data they could steal would be the entire product catalog. However, this information could be gained without any attack by simply systematically going through the entire pub-

[17]Crabb, J. "The BestBuy.com Cloud Architecture", IEEE Software, March/April 2014, pp. 91-96

lic website. The backend which processes actual customer purchases is obviously much more critical, as it has to store customer information such as credit card data and addresses, the loss or disclosure of which would be much more impactful. Hence, it makes sense to keep this part of the application in the internal data center, where the organization has full control over whatever security mechanisms are desired.

Irrespective of which hybrid architecture is used, the question remains how the combined use of private and public cloud infrastructure can be achieved in practice. One possible approach is to use in-house components provided by a public cloud provider. For example, Amazon's AWS offers so-called AWS Outposts, which essentially deploy parts of the AWS Infrastructure in the in-house datacenter of customers[18]. Similar functionality is available through Microsoft's Azure Stack[19] and Google's Anthos[20] platform. Another way to implement a hybrid cloud is to use an open source cloud platform such as OpenStack[21] in house, and to scale out into a public cloud provider that uses OpenStack as well[22]. A third possible approach that allows organizations to combine private and possibly even multiple public clouds together is to build applications on top of a platform that is supported across different clouds, such as the container orchestration platform Kubernetes[23]. This platform will be explored in more detail in Section 5.3.4.

5.3 Architectural Building Blocks for Cloud Applications

Cloud applications can be built on top of different computational platforms. This choice has far-reaching implications for the whole application life cycle, as well as for the degree to which the different goals described in Section 5.1 can be attained.

[18]https://aws.amazon.com/hybrid/
[19]https://azure.microsoft.com/en-us/overview/azure-stack/
[20]https://cloud.google.com/anthos
[21]https://www.openstack.org/
[22]https://www.openstack.org/marketplace/public-clouds/
[23]https://kubernetes.io/

5.3.1 Infrastructure as a Service

The most "traditional" way to use cloud services is to simply use the cloud as a virtualized data center, very much following the Infrastructure as a Service (IaaS) model. One of the main advantages of this approach is that it makes the migration of existing applications into the cloud very easy. Rather than redesigning the application with a different architecture, it can be moved to the cloud mostly as-is, by replacing the existing physical in-house server infrastructure running the application with a 1:1 copy of virtual machines in the cloud.

Such a move can yield many immediate benefits, such as increased system reliability (due to better fault-tolerance of the cloud infrastructure) and possibly cost savings due to the economics of scale by the cloud provider. However, several of the potential benefits of cloud computing cannot be fully achieved with such an approach. For example, consider an organization that runs its website in-house. A typical architecture for this would be to use a load balancer that distributes incoming requests between a fixed number of web servers. The scalability of such a setup is limited by the number of web server machines. The same will be true if the application is moved to the cloud by simply moving each individual server into a corresponding virtual machine. Hence, the ability of a public cloud to offer virtually unlimited automatic scalability is not realized.

In order to take full advantage of the possible benefits of cloud computing, a more cloud native approach than pure IaaS is needed. While there is no generally accepted definition of the term, its meaning is fairly well expressed by the following: "Cloud native infrastructure is infrastructure that is hidden behind useful abstractions, controlled by APIs, managed by software, and has the purpose of running applications"[24]. In other words, to achieve the scalability that cloud computing can offer, traditional IaaS has to be supplemented by some higher-level abstractions offered by the cloud provider.

In the Google Cloud Platform, the cloud-native abstractions of Instance Groups[25] and Load Balancers[26] can be used to achieve automatic scalability. While a detailed description of both concepts is beyond the scope of this book, we will provide an overview of how they can be used in the example of the company that is moving its web servers from in-house hosting into the cloud. The first step would be to define the web server instances

[24]Justin Garrison, Kris Nova: Cloud Native Infrastructure, O'Reilly, 2017

[25]https://cloud.google.com/compute/docs/instance-groups

[26]https://cloud.google.com/load-balancing/docs/https

through an Instance Template, which specifies the underlying virtual machine type and its resources, the boot disk image to be used, a startup script etc. An Instance Group is then configured to manage a number of instances based on the underlying Instance Template. Many detailed settings are available. Some of the most important ones include the minimum and maximum number of instances that should be maintained, the type of metric that should be used to determine load (such as requests per seconds or CPU load) and the thresholds above and below which the number of instances should be increased or decreased, the type of health check that should be performed to determine whether instances are still functioning (e.g. one ping per minute) etc. Finally, a Http(s) Load Balancer is attached to the Instance Group. Once again, a wide range of available settings can be used to specify how the Load Balancer should distribute incoming client requests between the different web server instances in the Instance Group.

Using an Instance Group and a Load Balancer, the system can relatively easily be configured so that a certain number of web server instances is maintained at all times (even during low load) to ensure reliability in the face of potential outages of individual instances. Similarly, it can rapidly and automatically respond to rising or falling numbers of client requests, ensuring low response times and cost effective operation.

While all the major cloud providers provide similar features, they are not standardized. In other words, using them makes switching between vendors a little harder than just sticking to pure IaaS would have. However, this disadvantage is in most cases be outweighed by the more powerful functionality. It should also be pointed out that while Load Balancers and Instance Groups clearly are "useful abstractions, controlled by APIs, managed by software [...]", i.e. can rightfully be considered cloud native features, building an application in such a way is still far from the most cloud native approaches possible. After all, the cloud user still bears the responsibility for the design of the underlying Instance Template, and is (through the configuration of the Instance Group) at least indirectly involved in the management of individual virtual machine instances. Some of the serverless approaches described in the sections below are more cloud native, as they reduce the administrative overhead and allow developers to focus more on the actual application and less on the infrastructure used to operate it.

5.3.2 Platform as a Service

All the major cloud providers offer a Platform as a Service (PaaS) product. For example, Amazon Web Services has AWS Elastic Beanstalk[27], Microsoft offers Azure Cloud Services[28] and Google provides the GCP App Engine[29]. While these platforms are certainly comparable in overall features and functionality, there are many small differences, and the lack of standardization across the different platforms means that the choice of one provider makes switching to a different one more difficult than it would be if classic IaaS was used. On the other hand, using PaaS also reduces the administrative overhead compared to IaaS, because PaaS allows serverless operation. In other words, developers of an application deployed on PaaS do not need to worry about allocating sufficient compute resources to ensure scalability, reliability etc. PaaS could also be considered more cloud native than IaaS, because it offers more powerful and higher-level abstractions to developers.

In the following, we will briefly look at what an application deployed on the GCP App Engine could look like. An App Engine application consists of one or more services. The exact structure of a service depends on the programming language used. Supported languages include Java, Python, Go, Ruby and PHP. For example in Java, a service has to extend the HttpServlet class, and requests to the service invoke the doGet() method of the service class.

Each service can have one or several versions. Different versions can represent different development stages of a service. Keeping old versions of a service around allows users to roll a service back to an earlier version if the most recent version causes problems. Multiple versions can also be used for testing purposes. For example, in a typical A/B testing[30] scenario some users are routed to version A of a service while others are presented with version B. By keeping track of the outcomes such as what percentage of users with each version end up performing desirable actions like making purchases later in the process, developers can determine which version of a service performs better.

Each version of a service can run one, several or no instances. A version without an instance would simply be an inactive version (which could still be reactivated by creating instances for it), while the number of instances

[27]https://aws.amazon.com/elasticbeanstalk/
[28]https://azure.microsoft.com/en-us/services/cloud-services/
[29]https://cloud.google.com/appengine
[30]https://en.wikipedia.org/wiki/A/B_testing

for active services depends on the load of the application. By default, this scaling happens automatically. If necessary, developers can influence the scaling behaviour, for example by defining the metrics that determine scaling (e.g. request rate or response latency), by specifying the thresholds at which scaling should occur for these metrics, or by other configurations such as setting a minimum number of instances that should run at all times.

Figure 5.3(source: Google cloud website [31]) shows the overall structure of App Engine applications. Each application has at least one default ser-

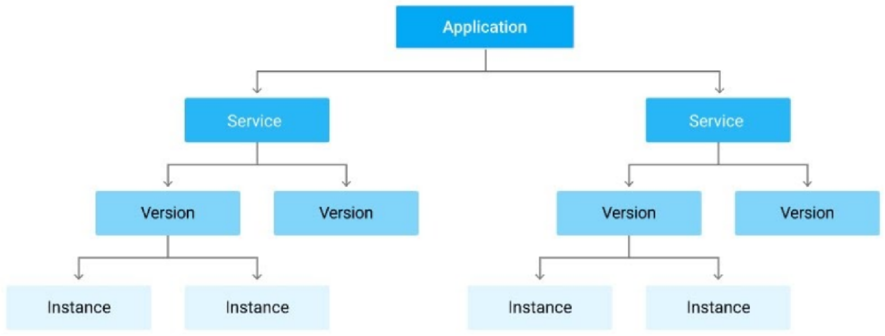

Figure 5.3: The structure of an App Engine Application

vice to which all incoming requests are routed by default. If the application has additional services, they can invoke each other using http requests. The URL of the request would include the identification of the requested service, and of the requested version of a service, if appropriate. User requests from outside of the application can be routed directly to the desired service either by including the correct service (and version) designation in the URL requested, or by creating a mapping based on patterns of the incoming URL to services in a dispatch configuration file. For example, all traffic from the mobile version of the application might be routed to a specific mobile service, while all other traffic is forwarded to a default service.

It is also possible to create services that run periodically without being triggered by specific incoming requests. Just like cron jobs in Unix-based systems, such services can be used to run regularly needed tasks such as daily backups or weekly status reports. All services in the AppEngine can also easily access other GCP functionality such as Cloud SQL, Storage or Pub/Sub to store data persistently (either in a relational database or in flat

[31] `https://cloud.google.com/appengine/docs/standard/python/an-overview-of-app-engine`

files) or to integrate asynchronously through messaging systems with other cloud components.

5.3.3 Functions as a Service

Just like with Platform as a Service, all the major public cloud providers offer Functions as a Service (FaaS) products as well. For example, Amazon Web Services offers AWS Lambda[32], Microsoft Azure features Azure Functions[33], and the Google Cloud Platform provides Cloud Functions[34]. FaaS is similar to PaaS in many ways - it is a serverless platform that can be considered to be rather cloud native, with all the advantages and disadvantages discussed above that this entails.

In contrast to PaaS, in which entire applications that can consist of a number of collaborating services are deployed together, the focus in FaaS is on deploying individual functions to the cloud. These functions should be stateless, which means that each invocation of a cloud function should not depend on previous or have influence on future invocations. There are many use cases for which FaaS is an appropriate choice. As an example, consider an application in which automatic processing needs to be performed on newly created data, for instance if users of an app can upload photos into a public location. In such a scenario, it would be a sensible precaution to automatically analyse uploaded photos to detect offensive or otherwise problematic content. To achieve this, a function performing such checks could simply be automatically executed whenever a new photo arrives.

Just like the GCP App Engine, Google Cloud Functions support a number of different programming languages including Java, Python, Go and Ruby. If Java is used, a cloud function simply needs to implement the interface HttpFunction, which contains a single method called service with a HttpRequest and a HttpResponse object as arguments. Cloud functions are connected to so-called triggers, which specify that a specific cloud function should be invoked when a specific type of event occurs. In addition to direct HTTP requests, cloud functions can be invoked by relevant events from Google Cloud Storage, such as the creation or deletion of a storage object, by the messaging service Pub/Sub, and by a few other services such as the mobile application development platform Firebase[35].

[32]https://aws.amazon.com/lambda/
[33]https://azure.microsoft.com/en-ca/services/functions/
[34]https://cloud.google.com/functions
[35]https://firebase.google.com/

GCP Cloud Functions can directly access a wide range of services on the Google Cloud Platform through API calls, ranging from different storage services to artificial intelligence based services such as the Cloud Translation or Vision APIs. Additionally, Cloud Functions scale automatically in response to the current load. For this purpose, the runtime automatically creates as many instances of the function as necessary to deal with all the incoming requests, and obviously scales the necessary underlying infrastructure accordingly. Different instances of the same Cloud Function, and different Cloud Functions are completely isolated from each other. All these features allow developers to focus on the intended functionality without having to worry about technical aspects such as managing and scaling the underlying infrastructure.

5.3.4 Containers

The idea behind containers in software architecture is similar to how containers work in the shipping industry. Prior to the invention and widespread adoption of containers, freight transported on ships or trucks was typically packaged in boxes or cases that were not standardized and differed widely between different kinds of goods. This made loading and unloading of transportation vessels rather time-intensive and at times difficult. The analogy in the software world is the difficulty of getting applications from the development environment deployed onto whatever target system they are intended to run on. This usually requires installation routines that are different for each application, and that have to ensure that the environment of the application, such as the required operating system, libraries, tools etc. all either exist or are created on a wide range of target machines.

Containers simplify commercial shipping significantly. A container ship can transport a certain number of containers, irrespective of their content, and loading and unloading can be done equally fast irrespective of what the different containers contain. In the software world, software that is packaged in a container can be deployed wherever containers are supported, and the container contains whatever environment the deployed software requires.

It should be pointed out that containers are not the only approach that simplifies packaging and deployment of software applications this way - the same can be achieved with virtual machines. Because virtual machines offer encapsulation, i.e. the entire state of a virtual machine is contained in a single file, applications can be moved from one machine to another through a virtual machine image. As long as the target machine for deployment runs the same virtualization software as the source system, the virtual machine running the application can simply be transferred as is. And since a single server can run multiple virtual machines simultaneously, different applications can simply be executed on the same server in different virtual machines. Of course, it is possible to combine VMs and containers together as illustrated in Figure 5.4 (source: WP website [36]): for example, a VM can run different containers.

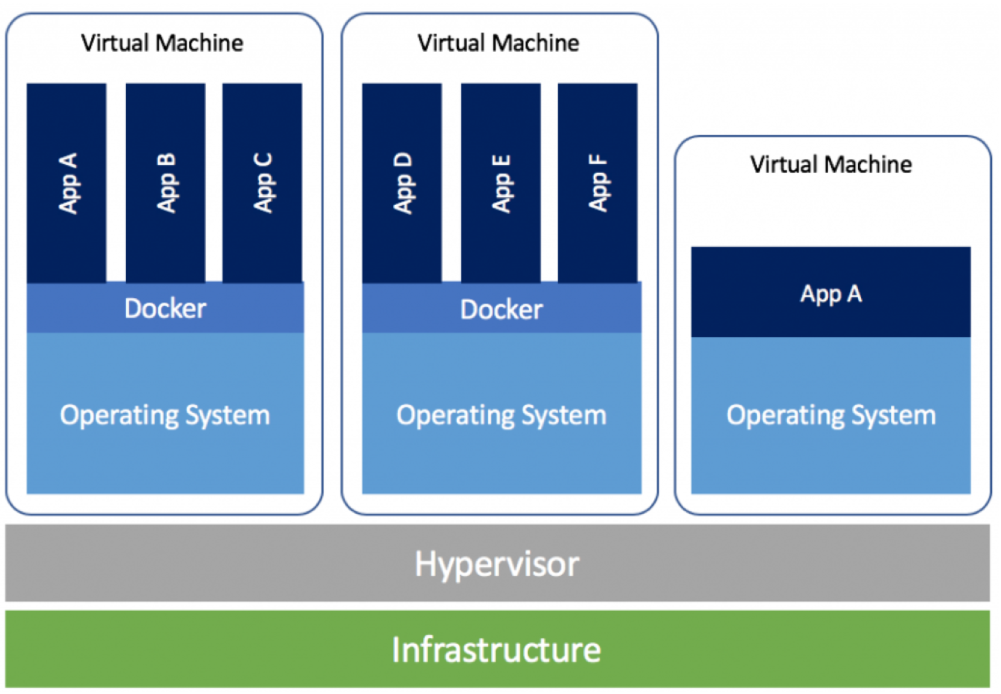

Figure 5.4: Containers and VMs can be combined together.

One of the main downsides of this approach is that virtual machines are rather heavy-weight. In order for software to work in virtual machines just like they would directly on physical machines, a virtual machine has to contain the full (virtual) hardware of a complete computer, along with a complete operating system with full drivers etc. This contains a lot of overhead, because the vast majority of applications do not require all the components of a complete virtual computer. Containers are ultimately just

[36]`https://i1.wp.com/www.docker.com/blog/wp-content/uploads/Are-containers-..-vms-image-2-1024x759.png?ssl=1`

light-weight virtual machines that contain everything that the applications deployed in them need.

While there are other container standards such as rkt[37] or LXD[38], Docker[39] is by far the most widely adopted container standard. Figure 5.5 (source: the Docker's website [40]) shows the overall architecture of Docker. A server

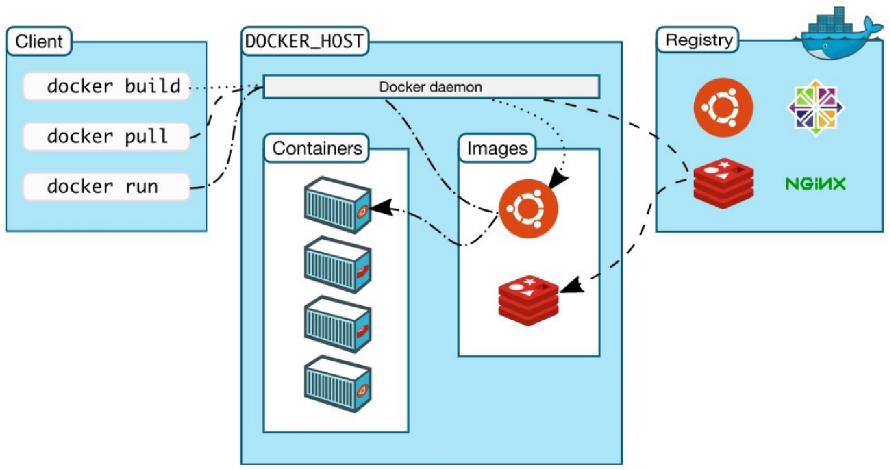

Figure 5.5: The Docker Architecture

running docker containers runs the Docker daemon (dockerd), a background process that manages all running containers. It is controlled through a client program called docker with which containers can be created and managed on the server. Docker containers are actually running instances of Docker images. In other words, a software developer who wants to distribute their applications creates a Docker image which can then be instantiated on any server running Docker. Docker registries are a convenient way of distributing or managing different Docker images, and can be either private to an organization or publicly accessible.

In practice, things often get more complicated than just deploying an application in a single instance of a container. For example, it is often desirable to deploy a single application in multiple containers, for example to be able to scale the application in response to increasing load. Especially if this deployment of containers occurs across multiple physical servers, the

[37]https://www.openshift.com/learn/topics/rkt
[38]https://linuxcontainers.org/lxd/introduction/
[39]https://www.docker.com/
[40]`https://docs.docker.com/get-started/overview/`

deployed application becomes significantly more fault tolerant, increasing overall reliability. Additionally, even before fault tolerance or scalability are considered, it is often appropriate to deploy the different parts of a single application in multiple containers. There are many good reasons for this. For instance, if an application consisting of a web server and a database is deployed in two separate containers (one for each component), each of the two components can be developed, updated and deployed more independently, which is especially advantageous in agile environments with a lot of change. Similarly, it may well be that one of the two components - e.g. the web server - becomes the performance bottleneck of the application. If that is the case, deploying the database in a separate container makes it possible to scale the web server independently from the database by simply increasing the number of container instances for the web server container, but not for the database container.

While such multi-container deployments can be managed by Docker directly, it has become common to use container orchestration tools that automate container management tasks. The most popular such tool is Kubernetes[41]. It is beyond the scope of this chapter to discuss Kubernetes in detail, but we will look at some of its key concepts. They include nodes, clusters, pods and deployments:

- Node: A node is the basic abstraction of computing hardware that Kubernetes manages. In other words, nodes could be physical servers, virtual machines or a combination of the two on which Kubernetes can deploy containers. Users do not directly interact with or manipulate nodes.

- Cluster: A cluster is a combination of nodes managed together by Kubernetes. Nodes can dynamically be added or removed from a cluster. When that happens, Kubernetes can adjust how applications are deployed accordingly.

- Pod: A pod is the smallest deployable unit, and can contain either a single or multiple tightly coupled containers. All the containers in a pod are scaled up and down together.

- Deployment: Deployments are the high-level abstraction through which applications can be managed. For example, a deployment can specify that a certain number of replicas of a pod should be created on

[41]https://kubernetes.io/

a cluster. This would mean that if a node in the cluster on which an instance of the pod was deployed failed (e.g. due to a hardware problem), Kubernetes would automatically create another instance of the pod on another node in the cluster to reach the specified number of replicas again.

Some of the key features of Kubernetes include:

- Automatic load balancing between pod instances.
- Automatic scaling by creating more instances in response to rising load, and by reducing their number when load decreases.
- Health checks of instances that can be used to automatically restart or replace non-responsive or failed instances.
- Automatic rollout and rollback of updates to the deployed containers to allow both for quick implementation of changes as well as the ability to go back to a previous version if an update causes problems.
- Support to securely configure infrastructure needs such as access to permanent storage, discovery of and communication between instances of services in other containers, and the management of secrets such as usernames and passwords used to access resources such as databases.

This rich set of features makes Kubernetes a compelling choice even in completely self-hosted, non-cloud applications that use a flexible and microservice style approach.

All the major public cloud providers offer Kubernetes deployments on their platforms - AWS provides the Amazon Elastic Kubernetes Service[42], Microsoft Azure has the Azure Kubernetes Service[43] and the Google Cloud Platform the Google Kubernetes Engine[44]. Using Kubernetes in the cloud offers the additional advantages of not having to manage the underlying hardware and infrastructure running the platform, as well as the ability to scale applications in a practically unlimited way. Furthermore, since Kubernetes is supported across a wide range of cloud platforms, using it reduces the risk of vendor lock-in, especially compared to other cloud-native approaches like Platform as a Service. The previous sentence says

[42]https://aws.amazon.com/eks/
[43]https://azure.microsoft.com/en-us/services/kubernetes-service/
[44]https://cloud.google.com/kubernetes-engine

"reduces" rather than "prevents" because despite the fact that different providers offer Kubernetes, different implementation details still mean that moving applications between different Kubernetes environments entails at least careful testing, if not some adjustments. Finally, since Kubernetes itself can flexibly integrate nodes from different platforms, it is relatively easy to build hybrid or even multi-cloud applications in which some nodes are hosted in an in-house data center while others are based e.g. on virtual machines on two different cloud providers.

In addition to the Kubernetes Engine which has the full feature set of Kubernetes and gives users control over all the different aspects of container orchestration, the Google Cloud Platform also offers Cloud Run, which is another way to run containers serverlessly in the cloud. Cloud Run is more fully managed by Google, which means it gives users less control over the details of deployments, and offers fewer advanced features. However, for simple stateless applications it might be the better choice since it requires even less user involvement in the operation of deployed applications.

5.4 Summary

Throughout this chapter, it hopefully became clear that there is not a single best cloud application architecture, but that there are a wide range of choices, each of which has different strengths and weaknesses in different situations. There are a number of factors that should ultimately decide the best cloud architecture for a given application, from the specific business needs and their influence on the relative importance of the different architecture goals to the existing IT infrastructure and expertise in the organization (because cloud applications are usually not built entirely from scratch either).

It should also be repeated that this chapter barely scratched the surface of this topic area - the interested reader should easily be able to find entire books on many of the sections and topics covered, be it concepts such as microservice architectures or specific products such as Kubernetes. Additionally, for many applications it makes sense to not build all the technical aspects from scratch, but to use frameworks or toolsets like Google Firebase[45] which takes care of commonly needed technical services such as authentication, messaging, storage etc., and consequently save developers

[45]https://firebase.google.com/

from having to “reinvent the wheel” for generic, not application specific tasks.

Chapter 6

Cloud Security

For reasons that will be discussed in more detail below, security is one of the biggest concerns in cloud computing. Since cloud computing is built on "normal" IT resources such as servers, networking equipment, storage etc., traditional IT security concepts apply in cloud computing as well. The first part of this chapter briefly reviews IT security. The second part looks at how cloud computing can both help and hinder achieving IT security. The final part describes best practices on how to improve security in the cloud, both in general and specifically in the Google Cloud Platform.

6.1 A Brief Review of IT Security

IT Security is often broken down into three concepts: Confidentiality, integrity and availability:

- **Confidentiality** describes that data is only accessible to whoever is authorized to access it, and nobody else. The importance of confidentiality can be illustrated using countless examples. For example, if credit card information of a person is disclosed to somebody unauthorized, it might be misused to make unintended purchases. Medical information is generally considered to be even more sensitive - most people so not want their medical history to be disclosed, not only for privacy reasons, but also because such a disclosure might lead to harmful outcomes, for example being turned down when applying for different types of insurance, not being hired for jobs due to health concerns etc.

- **Integrity** means that data cannot be corrupted or otherwise changed by unauthorized users. For example, accounting records reflecting sales transactions are generally not allowed to be changed. If errors are discovered later, a correcting transaction should be recorded. Similarly, student grades should only be changed through a proper process if errors are discovered, and not because a student manages to break into the database and decides to improve their grades.

- **Availability** requires that data and/or applications that use the data is actually accessible and working properly for their users. At first sight, it might seem that availability is not really related to security, or at least that it is less important. But two main points can be made to illustrate the importance of availability: Firstly, without the requirement of availability, confidentiality and integrity are trivially easy to achieve: If data is stored on a hard drive and locked up in a safe, nobody can steal or modify any of it. Secondly, and probably more importantly, computerized systems are often of critical importance for the organizations that use them. An online retailer with a website that cannot be accessed by its customers, a bank that cannot perform transactions, or an airline that cannot access its flight booking systems (either to check passengers into flights or to sell seats for future flights) incur such massive financial damage by the unavailability of their systems that the survival of the company may be at risk if an outage is not resolved in a matter of hours. Patient care in a hospital that cannot access patient data (such as diagnoses, laboratory results, treatment plans) could suffer to a point where the health or even life of patients are at risk.

There are of course many other categorizations of IT security such as the STRIDE[1] model used by Microsoft, which is an acronym of the terms Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. Tampering is equivalent to violated integrity of data, information disclosure corresponds to a lack of confidentiality and denial of service is an attack on the availability of data or systems. The other elements of STRIDE describe more specific ways to attack systems to achieve one or several of these outcomes. The following sections review two more comprehensive categorizations of IT security risks.

[1]`https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats`

6.1.1 Common IT Security Risks

IT systems face a wide range of risks and possible attacks that endanger one or several aspects of security (confidentiality, integrity and availability). A relatively simple classification scheme by Jouini et al[2]. distinguishes the following factors:

- Threat source: Either internal (e.g. an employee) or external (e.g. an outside attacker)
- Threat agents: Human actors, environmental factors or technological factors
- Threat motivation: Malicious or non-malicious
- Threat intention: Intentional or accidental
- Threat impact: Destruction of information, corruption of information, disclosure of information, denial of use etc.

Each real-world threat can be viewed as a combination of these five factors. For example, an internal human employee could non-maliciously and accidentally destroy data by accidentally deleting it with an incorrect shell command. Or, an external human hacker could maliciously and intentionally break into a system to steal personal information of customers of a company in order to later commit identity theft. Or, an external environmental factor (a massive rain storm) could accidentally flood a server center, making the applications and data hosted in it temporarily inaccessible. Clearly, many different combinations of the five factors are possible, indicating that there is a significant number of different threats to IT security. Herzog et al. [3] classify specific threats with Figure 6.1.

It is beyond the scope of this book to discuss all these threats in detail. Instead, we only highlight a few examples:

- **Buffer overflow attacks:** If programmers do not carefully check user input, attackers can use unexpectedly long values in input fields to write data beyond the memory locations allocated for variables. If this attack is executed skillfully, attackers can write executable code

[2]Jouini, M., Rabai, L.B.A., Aissa, A.B.: "Classification of security threats in information systems", 5th International Conference on Ambient Systems, Networks and Technologies, 2014

[3]Herzog, A., Shahmehri, N., Duma, C.: "An Ontology of Information Security", International Journal of Information Security and Privacy, January 2007

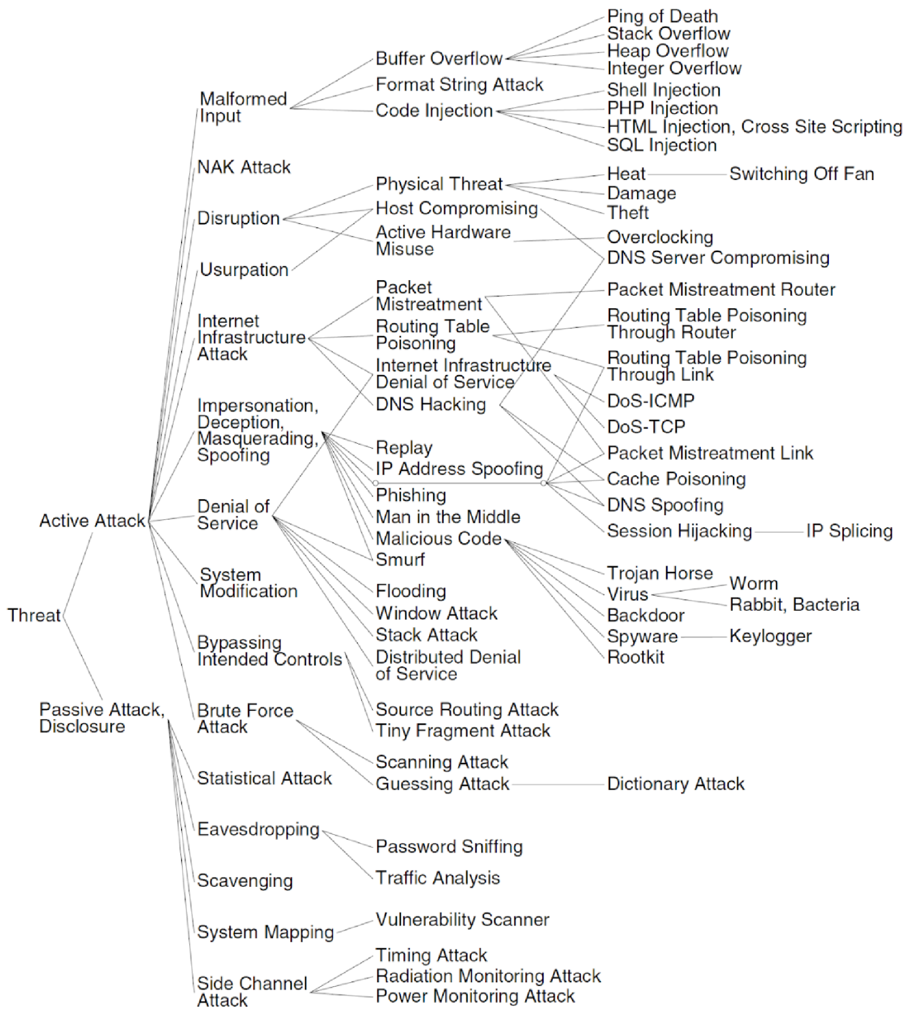

Figure 6.1: Classification of IT Security Threats

into memory and trick the runtime to execute it, which enables them to take over control.

- **Cross Site Scripting:** Similarly, if user input to online forums or message boards is not checked carefully, attackers can insert malicious script code that is then executed in the browser of other users.

- **SQL Injection:** Yet another related attack, in which a database is tricked into executing malicious SQL code by specially prepared (and insufficiently checked) user input, typically in web forms.

- **Physical theft of hardware:** Attackers can steal end user devices (such as laptops or smartphones) or servers (or simply server hard drives) to get access to confidential data.

- **Phishing attacks:** Attackers trick users into revealing confidential data, such as their login name and password to an online banking site, by sending them emails that look like they come from their bank and lure them to a fake (but real looking) bank website.

- **Man in the middle attacks:** In this type of attack, users think that they interact with for example their online banking website, when in fact they are communicating with a website set up by an attacker that looks like the real website, and that forwards user requests to the real website and displays its results back to the end user in order to conceal the existence of the attacker.

- **Malicious code such as viruses:** A common type of virus is ransomware, which encrypts important data (documents, photos etc.) in the background, and then asks users to pay a ransom to get access to the decryption key, or lose all their data otherwise.

- **Distributed Denial of Service attack (DDoS):** In a DDoS attack, a large number of computers (often without knowledge of their owners, because they have been taken over by attackers) flood a service such as a website with so many fake requests that the website becomes completely overloaded and cannot respond to real user requests anymore.

- **Eavesdropping:** Attackers intercept network packets, e.g. in a Wifi network, for example to intercept user passwords for websites.

- **Vulnerability Scanner:** Attackers systematically scan open ports of a server for installed software to find known vulnerabilities (due to missing security updates) that can be exploited for other attacks.

6.1.2 General IT Security Measures

When many of the IT technologies and standards that are still used today were developed, security was not a major concern. For example, many of the networking standards still in use today are from a time when only a small number of participants (typically researchers) who generally considered each other to be trustworthy, or at least not malicious, used them. Additionally, in the early days of what we call the Internet today, there was very little that a potential attacker could gain - no payments or purchases were made over the network, no medical or other sensitive data was stored on reachable servers etc.

As technology advanced and the Internet became used more and more for commercial purposes, the need for security increased, and an ever growing market for security software and services began to emerge. The marketing approach of many IT security vendors (such as Anti-Virus software companies) seems to be to try to convince their prospective customers that if they adopt their security product, security will be a solved problem. While this is a very comforting message, it is unfortunately very untrue. There is no single measure anybody can take to secure their IT systems. Instead, IT security is often pictured as a metal chain which is only as strong as its weakest link.

IT security includes (at least) the following aspects:

- Host security
- Network security
- Encryption
- Access control
- Human/organizational aspects

To illustrate the previous point, even if all the host machines and the network are very secure, everything is encrypted and access control uses strong credentials, all it takes for security to fail is a user who falls for a social engineering attack and gives out their username and password to an attacker. Or, if users are well trained to not fall for such attacks, and everything else is very secure, but an attacker pretending to be a technician can gain physical access to the server room, such an attacker can simply steal or destroy storage media with sensitive data right out of the servers, once again breaking security.

The following gives a very brief overview of different aspects of IT security:

- **Host security** involves securing the individual server and clients machines that are used. This begins with physical security, which means that devices cannot be stolen or otherwise accessed by unauthorized users. Also, hosts should be hardened against potential attacks, which involves disabling or uninstalling any unnecessary services to reduce the potential attack surface, to install all available security patches, and to include appropriate controls in all application software to e.g. prevent buffer overflow attacks. Finally, regular backups of important data should be performed, to ensure availability of data in case of successful attacks or simple hardware failures.

- **Network security** involves partitioning networks into subnets to separate hosts that are in different security domains. For example, servers that need to be reachable from outside of the organization should be in a different subnet than client machines that should be hidden from the outside world. Between the different network segments firewalls should be installed. These firewalls check all network traffic and only allow authorized traffic to pass through. In addition, intrusion detection and/or prevention systems (IDSs/IPSs) should be installed. Both IDSs and IPSs constantly monitor network traffic, and trigger an alarm (in the case of an IDS) or even block suspicious network traffic (in the case of an IPS) if unusual traffic occurs.

- **Encryption** is often considered a separate security measure different from host security and network security measures. Both moving data (i.e., network traffic) and resting data (stored data) should be encrypted for security purposes. This prevents attackers who might succeed in eavesdropping on network traffic or get physical access to storage media from actually getting access to confidential data. One of the challenges of encryption is the management of encryption and decryption keys- attackers must be prevented from getting access to these keys, because otherwise the whole exercise of encryption becomes pointless. On the other hand, keys can also not be lost, because otherwise the data becomes unusable even to authorized users.

- **Access Control** describes measures to ensure that users can only access whatever data or resources they are authorized to access. This usually involves authenticating users (i.e. enabling users to prove their identity to the system) and keeping track of what resources an individual user is actually authorized to access. The first part is usually still done with usernames and passwords, even though this approach has many problems, some of which can be mitigated by multi-factor authentication, where a user does not only have to know a password, but also has to be in possession of e.g. a specific hardware device to access the system. The second step, determining the permissions of an authenticated user, is often done with constructs such as roles, which group users that share the same permissions together.

- **Human/organizational aspects** must be considered since IT security is not a purely technical problem, and consequently cannot be achieved by only technological solutions. In particular, as IT systems are used by individual users in organizations, appropriate security policies and procedures have to be in place. For example, the best encryption, the strongest passwords etc. can be defeated if an attacker can convince a user to share their password with them. Similarly, if a user copies important data onto a USB drive so that they can take it home to continue working there, simple loss or theft of that drive can break the confidentiality of that data. Therefore, organizations need to reduce the likelihood of such occurrences by appropriate policies, procedures and constant security training and awareness measures.

6.1.3 IT Security Limitations and Tradeoffs

As the brief overview of IT security risks and security measures above has probably made clear, IT security is a highly complex issue. One of the consequences of that is that achieving complete security is practically impossible.

IT security involves several tradeoffs. In particular, security can generally be increased by spending more money and effort. This relationship is not surprising - spending money on additional security products, hiring more security experts etc. is generally going to increase security. Another important tradeoff is between security and user friendliness - the higher the level of security required, the less user friendly systems tend to be. Access control is a good example - security enhancing requirements like long and complex passwords, frequent changes of passwords, multi-factor authentication etc. make life harder for users.

It follows from these tradeoffs that complete security is not only impossible, but often also not even desirable. Instead, each organization needs to assess its risks and security needs and consequently balance its security measures accordingly. For example, for a hospital processing personal health data about its patients, any unauthorized disclosures of that data, or even temporary unavailability of that data could have very severe consequences, in extreme cases up to the death of patients. Accordingly, such an organization should strive to achieve a high level of IT security, even if it comes at the expense of significant cost and reduced comfort for its users. On the other hand, an amateur softball league might run a website on which upcoming matches and past results are posted. While a temporary unavailability of such a website or the manipulation of its data might be annoying, it would clearly have significantly less severe consequences compared to the hospital example. Therefore, it does not make a lot of sense for the softball league to heavily invest in IT security because of the additional cost, which ultimately has to be paid for by league members who would gain little in exchange for the expense.

There is a fairly popular joke that can be found on T-shirts aimed at tourists in many places in Canada: "You don't have to outrun the bear, you only have to outrun the person with whom you are walking in the wilderness". Applied to IT security this means that an organization (or individual) only has to strive to make their IT more secure than many of their peers. If that is achieved, many kinds of security risks are significantly reduced, because attackers are likely to "go for the low hanging fruit". For example, a criminal organization with the business model to infect people's computers with ransomware generally goes for the least well defended victims. A common attack vector is to scan the systems of potential victims for known security vulnerabilities that can be exploited to infect them. As long as a significant percentage of users does not regularly update their operating systems, browsers, browser-plug-ins etc., those who do are not necessarily safe from a ransomware attack, but probably have a much reduced risk.

Of course, this approach is not sufficient for larger organizations or organizations that handle particularly valuable data, because in such cases attackers might be willing to spend the extra effort necessary for a successful attack due to the expected higher return. Therefore, even if such organizations "do their homework" and implement basic IT security measures, they can still be attacked successfully. In the absence of gaping security holes, successful attacks typically involve multiple stages, which might include social engineering attacks on employees to get a first foothold in the organization e.g. by taking over the computer of an employee. The attackers then progress step by step towards the actual goal of the attack (e.g. by attaining local administrative rights, then gaining access to important servers, taking over networking equipment to open a channel to transfer data out etc.). These multi-stage attacks typically take several days or even weeks to succeed. The concept of Mean Time To Breach (MTTB) is used to indicate how long a successful attack takes on average. While it is impossible to attain the exact value for MTTB in an organization, the analysis of past attacks, or "read team" approaches where a team of outside security consultants is paid to attempt to break into IT systems can be used to get an approximate value. A second important security concept is that of Detection Coverage, which expresses which percentage of hacking activities is discovered by the intrusion detection systems of an organization. The goal is to detect attacks before they achieve their final goal. Consequently, the lower the MTTB is, the more effort an organization should put into achieving a high Detection Coverage, to increase the chance that an attack can be discovered and ultimately foiled.

6.2 Impact of Cloud Computing on IT Security

If cloud computing is used, the responsibilities for the various IT security measures are split between the cloud service provider and the cloud service user. How exactly they are split depends heavily on what type of deployment model and service model is used. For example, an organization that uses Infrastructure as a Service in a private or hybrid cloud model has to spend a lot more effort on in-house IT security than an organization that uses Software as a Service with a public cloud service provider. In the latter case, the provider takes care of most of the host and network security measures, while the cloud service user is mostly concerned with human and organizational security measures.

Irrespective of the deployment and service models used, a common question is whether cloud computing is secure. This question is addressed brilliantly in a paper entitled "Is Chocolate Good for You - or, Is the Cloud Secure?"[4]. The authors make the point that the first question can only be answered by "it depends" - on what is meant by "good for you" (health, well-being, happiness, etc.), on exactly what kind of chocolate is consumed

[4]Netkachova, K., Bloomfield, R. "Is Chocolate Good for You - or, Is the Cloud Secure?", IEEE Computer, August 2017, pp. 74-78

in what quantity and frequency, by characteristics of the individual consuming the chocolate etc[5]. In analogy, cloud computing is neither secure nor insecure in general - it depends on what type of cloud services are used, what the cloud deployment and service models are, what security needs the organization has, what specific security measures are taken etc. It is generally accepted that cloud computing introduces or increases certain risks, especially if not only in-house private cloud computing is used. However, cloud computing also offers several opportunities to reduce security risks (see Derrek[6] and Jansen[7] publications). This section discusses both these risks and opportunities in turn.

6.2.1 Cloud Security Risks

This section describes risks that might be introduced or worsened by the use of cloud computing.

System Complexity
Cloud computing environments are generally very complex. Workloads usually do not run directly on hardware, but on top of a virtualization layer. Similarly, storage and network resources are also typically virtualized on top of physical hardware. The cloud service provider has to operate complex management systems to allocate resources efficiently, to bill customers in a very fine-grained way, to implement automatic scalability and failover, to allow the geographic distribution of resources etc.

This complexity not only increases the attack surface compared to systems that are run directly on physical hardware, it also makes accidental configuration errors that can be exploited for attacks more likely. On top of that, the incentive for attackers to find security holes in a cloud service provider is very high, because they represent a very attractive target: A successful attack on a public cloud service provider might give access to data from multiple customer companies.

[5]The author feels he would be amiss if he did not express his unscientific (yet strongly held) opinion that the regular consumption of a reasonable amount of quality chocolate is in fact good for most people.

[6]Dekker, M.A.C., Liveri, D.: "Cloud Security Guide for SMEs", European Union Agency for Network and Information Security, April 2015

[7]Jansen, W., France, T.: "Guidelines on Security and Privacy in Public Cloud Computing", National Institute of Standards and Technology, special publication 800-144, December 2011

Multi-Tenant Environment
Public cloud providers usually utilize multi-tenancy, i.e. physical hardware is shared by multiple different customers. For example, virtual machines of multiple customer companies can be placed on the same physical server. The obvious benefit of this approach is that the cloud service provider can utilize its hardware resources more efficiently, which reduces costs. In theory, the different tenants on the same physical hardware are strictly isolated from each other. In practice, this isolation can break down due to bugs in software such as virtualization software or even hardware such as modern CPUs. The most famous example for the latter are the Meltdown and Spectre vulnerabilities[8] that affect many server CPUs.

Multi-tenancy not only introduces the risk that confidential data might be accessed by unauthorized parties, it also might affect availability of resources. Even though modern virtualization platforms offer performance isolation features that (once again, in theory) guarantee that resources such as CPU time, memory or network capacity cannot be monopolized by one of the virtual machines running on a physical server, these guarantees might break down in the face of implementation errors or extreme situations such as distributed denial of service attacks.

System Management is Internet-Facing
In applications that are hosted in a traditional in-house server center setting, there is a strict separation between the customer-facing functionality that is accessible from the Internet, and the system management functionality that is only accessible from the internal network and shielded from the outside Internet through firewalls, often in a *demilitarized zone* (DMZ) configuration. If a public cloud service provider is used, this separation is not possible, since the management interfaces of the cloud platform are also accessed through the Internet. This significantly increases the potential attack surface.

Vendor Lock-in
Vendor lock-in is a particularly important issue in cloud computing because the majority of services offered by the different providers are proprietary, despite being similar in functionality. This is a security issue because it potentially puts the availability of services that are hosted by a public cloud provider at risk, for example if the provider goes out of business or if there are significant contractual disagreements with the provider. To

[8]https://meltdownattack.com/

mitigate the risks of vendor lock-in, multi-cloud systems have been implemented as discussed in Chapter 7.

Legal Issues
Legal risks with cloud computing are often caused by a combination of other risks, such as multi-tenancy and vendor lock-in. For example, legal problems of other customers who happen to share the same physical hardware might result in court-ordered actions such as confiscation of hardware that could affect uninvolved parties. Another significant issue is that due to the global nature of large cloud service providers, foreign laws might apply to applications that are hosted in different geographic locations. For example, content that is allowed by free speech laws in one country might be illegal due to anti-defamation, youth protection or public decency legislation in other countries, potentially exposing companies who host their application across different jurisdictions to legal challenges that could affect their availability. Another common problem is that different jurisdictions (such as the European Union, but also Canada) have stricter privacy laws than other jurisdictions (such as the USA), potentially putting organizations that transfer data into countries with lower standards into trouble with government regulators.

6.2.2 Cloud Security Opportunities

The appropriate use of cloud computing can also increase IT security. This section briefly introduces some of the reason for that.

Elasticity
The elasticity offered by the large cloud service providers greatly increases the availability of an application that is hosted on it. A simple example could be the web site of a small online store. Before cloud computing was widely adopted, it was not uncommon that unexpected spikes in users (e.g. because the site was mentioned on a popular website or even TV show) effectively crashed the website, because the server could not keep up with the increase in requests. An even more serious scenario is a distributed denial of service attack, in which an attacker uses a large number of (typically hijacked) computers to flood a server with so many requests that it becomes unresponsive to real users. It is very difficult for an organization that hosts its applications on its own servers to defend itself against such attacks. In contrast to that, the use of cloud computing allows organizations

to set up their systems in such a way that any spike in demand (whether caused by legitimate users or a DDoS attack) is met by an automatic and near-instantaneous scaling up of capacity.

Geographic Spread
Public cloud service providers operate data centers in multiple geographic locations. While this has other benefits such as lower latency for end users, it can also be used to increase the availability of an application. Even though modern data centers (such as the ones operated by cloud service providers) go to great lengths to protect against service outages, they can never eliminate all risks. For example, data centers typically have at least two redundant Internet connections and emergency power generators in case of power failures, in addition to significant physical protections. However, for instance sufficiently severe natural disasters such as earthquakes can still render such a data center inoperable. Therefore, an organization that hosts its applications across multiple geographically spread-out data centers of a cloud service provider can virtually ensure the availability of its applications in the face of natural disasters, as long as they are not of a global nature. This benefit not only applies to applications, but also to backups of data, which, unless stored in multiple geographic locations, are also at risk of destruction in severe disasters such as large-scale fires.

Economics of Scale of Cloud Service Providers
The sheer size of the large public cloud service providers is an additional factor that can increase security significantly. For example, data centers operated by cloud service providers are likely significantly better protected from physical intrusion than self-hosted servers at most organizations. Economics of scale of cloud data centers (which typically contain at least tens of thousands of physical servers) allow providers to split expenses for physical security over thousands of customers. Therefore, they can afford to spend significantly more on physical security, which results in not just higher availability, but also higher integrity and confidentiality of customer data and applications.

Essentially the same argument applies in other areas that are relevant for security. Cloud service providers are likely able to have more, more specialized and better security exports than any customer organization could employ on its own. Consequently, they are likely able to respond quicker and more effectively to security incidents, and can more extensively test and more quickly install available security patches and updates, again increasing all aspects of security.

Server-Side Storage
Cloud-based applications typically store data not on client machines, but in the cloud. The contrast can easily be illustrated with an office application suite containing a word processor, spreadsheet, presentation and database applications. Traditionally, such application suites were installed on end-user machines, and kept user data (documents, spreadsheets etc.) there as well. If such an application suite is hosted as a SaaS offering in the cloud, then both the applications and all user data are stored in the cloud, while client machines only need a web browser and an Internet connection. An example for this would be the so-called Chromebooks, which are laptops powered by Google's Chrome OS and which primarily rely on software and user data in the cloud. A significant security advantage of the cloud approach is that the loss or theft of end user devices such as laptops or tablets puts the confidentiality of data much less at risk, as long as user authentication is sufficiently robust.

Certification and Compliance
A final security-related point is related to regulatory requirements. Governments require companies operating in certain sectors (such as health care, financial services etc.) to demonstrate that they are complying with very specific regulatory requirements, some of which are about IT security. The reason behind these requirements is that for instance IT outages at organizations like credit card companies do not just affect those companies themselves, but just about any other company (and countless consumers) who use credit cards. To help protect the broader economy from significant damage due to negligence of for example credit card companies, they are required to take specific and significant steps to protect their systems.

In order for companies that have to fulfill such compliance requirements to be able to use public cloud service providers, the providers have to fulfill the compliance requirements themselves. This gives anybody else who uses public cloud services additional assurances about their security. It is also very efficient, since it allows countless customer companies to rely on the certified compliance of cloud resources they use, rather than having to audit compliance of in-house operations themselves.

It should be noted that the cloud security opportunities discussed here are exactly that - opportunities. Incompetent use of cloud services can easily negate these potential benefits. For example, a severe thunderstorm caused power outages that took down a large AWS data center in Virginia, USA for well over an hour[9]. Several popular websites, including Netflix, Instagram and Pinterest experienced service outages for many of their users, because they apparently hosted most of or all of their servers in this one AWS data center, rather than taking advantage of the availability of multiple geographically distributed data centers by the provider.

6.3 Cloud Security Best Practices

This section describes best practices for cloud security in a number of different areas, including access control, encryption, network security and monitoring and logging.

6.3.1 General Considerations

Before the best practices in these areas are covered, some general considerations about cloud security measures are presented.

Follow General IT Security Best Practices

The first best practice to achieve security in cloud computing is to follow general IT security best practices. Since the cloud is made up of "normal" IT resources (servers, networking equipment, storage etc.), a first important step towards securing the cloud is to secure the individual resources as discussed above, by installing all available patches and security updates, reducing the attack surface by only installing what is needed, ensuring physical security from intruders who might try to directly access or steal hardware etc. It depends on the cloud deployment and service models chosen to which degrees this falls into the responsibility of the cloud service provider and the cloud service user, respectively.

Choose Appropriate Deployment and Service Models

Related to this point, another important consideration is to choose appropriate deployment and service models. These choices depend highly on the situation. In one extreme, for particularly security critical applications, cloud computing might not be an appropriate choice at all - for example,

[9]. https://venturebeat.com/2012/06/29/amazon-outage-netflix-instagram-pinterest/ (accessed on July 7, 2021)

the control systems of a nuclear power plant should probably not be reachable over any network at all. In slightly less extreme cases, where particularly sensitive data is handled, and where government regulations might be very restrictive, private cloud use might be the only practical solution.

While the choice of service model depends on other factors as well, security considerations should play a big role. Given that on the spectrum from Infrastructure as a Service over Platform as a Service to Software as a Service the percentage of the security responsibilities increasingly shift from the cloud service user to the cloud service provider, users considering models further on the left (such as IaaS) should make sure that they are able to keep up with the necessary security tasks. To illustrate this with a simple example - an organization could host their website using IaaS on a cluster of virtual machines that run a system such as Wordpress on top of the LAMP (Linux, Apache, MySQL and PHP) stack, or they could use Wordpress as a SaaS solution offered by some provider. The former approach would give the organization more control over the infrastructure, but it would also make them responsible for keeping the whole platform - the Linux operating system, all the necessary packages including the web server, database, and Wordpress itself - up to date with security patches. This in turn requires personnel that is aware of security updates and carefully tests and installs them on short notice. Because employees sometimes go on vacation or get sick, this typically requires at least a small team of system administrators. Consequently, small organizations might not be able to guarantee timely installation of patches, or it might simply not be economical to keep the necessary in-house IT experts to do so, in which case a different service model like SaaS might be more appropriate from a security perspective.

Mitigate Cloud Security Risks

The next piece of general advice is to mitigate the security risks introduced or worsened by cloud computing as much as possible. In public cloud computing, dealing with the high degree of system complexity is mostly the responsibility of the cloud service provider. However, if private or hybrid cloud computing is used, extra care should be taken to not inadvertently open security holes by careless configuration errors.

Similarly, mitigation of risks caused by multi-tenancy is also mostly the responsibility of the cloud service provider. Even so, users can take extra measures with particularly sensitive data, including the use of modern in-memory encryption approaches.

To reduce the impact of the fact that the system management interfaces are accessed through the Internet, extra care should be taken with how cloud administrators connect to the provider. This can include the use of Virtual Private Network (VPN) connections between the administrator's system and the cloud.

Finally, vendor lock-in and legal risks can be addressed by making sure that sufficient in-house IT expertise remains to not be completely dependent on the cloud service provider. The details depend very much on the situation, but organizations should always have an understanding of how they can export important data back from the cloud service provider, and how they could move to an alternative way to run their core applications, be it in house, or with a different service provider. Depending on the risk assessment, it might be worthwhile to follow a multi-cloud strategy, in which more than one provider is used, which greatly reduces the dependency on any single provider. Specifically with regard to potential legal problems, legal experts (either in-house or by external service providers) should be involved when cloud computing strategies are developed or changed.

Take Advantage of Cloud Security Opportunities

As discussed above, the security opportunities that cloud computing offers have to be deliberately exploited in order to be fully realized. While cloud computing promises practically unlimited scalability, this elasticity is only the result of appropriate application design and/or configuration. For example, if an application is deployed using IaaS, a properly configured auto-scaling load balancer can be used to ensure that the application smoothly scales up and down with system load.

Similarly, the geographic spread of available data centers by the large public cloud service providers has to be utilized in order to yield the increased resiliency against system outages caused by natural disasters. This can be realized by making sure that the application is deployed redundantly across multiple data centers.

Taking advantage of server-side storage, and leveraging the resulting reduced security impact of lost or stolen end-user devices requires applications that in fact do not store significant amounts of data on client machines, that ideally encrypt any data stored there, and that take appropriate steps to make sure that stolen devices cannot easily be used to access data on servers.

The security opportunities created by the economics of scale of the cloud-service providers and the ability to leverage certification and compliance steps taken by cloud service providers generally do not require specific actions by cloud service users to apply.

6.3.2 Access Control

Managing access control properly is a crucial component of IT security. It is especially relevant in a cloud computing context, because irrespective of the cloud deployment and service models chosen, making sure that the right users have the right permissions is a key task that has to be managed (at least in part) by cloud service users.

Access control is generally done with user accounts that have specific sets of permissions to access specific resources (such as servers, applications or storage). User accounts are not just given to real persons, but are also used for technical resources. For example, a program such as a web server typically runs with a specific user account on the operating system level that has specific permissions attached to it, such as the permission to read from and/or write to specific folders in a file system.

General Access Control Principles

All users should be authenticated using strong credentials. As far as human users are concerned, this should preferably require 2-factor authentication, especially for users who have administrative rights.

The principle of least privilege means that each user (whether a human user or a technical user) should have exactly the required permissions, and nothing more. For example, in a (oversimplified hypothetical) example where a single physical machine hosts a web server and a database server that is accessed by the web server to generate dynamic web pages, using just one technical account to run both processes would violate this principle. The web server might only need direct read access to the document root directory, write access to some logging directories and the permissions to send specific queries to the database. The database might only need read and write access to the storage locations of the database and write access to the same logging directories. Consequently, both processes should be run using separate accounts with exactly these permissions. This increases security in multiple ways. For example, if an attacker successfully takes over one of the processes, they cannot directly access data that is only used by the other process, increasing confidentiality and integrity of data. Similarly, programming errors in one process could not accidentally harm the data of the other process, increasing overall availability.

Another important principle is that of the separation of duties. In general, the idea is that no single user should be able to complete critical functions alone. A common example from the business world is that performing and recording transactions should be done by separate entities. For example, in a retail store, if the only record of sales transactions are recordings manually made by cashiers, then it becomes very easy for cashiers to simply not record a sale and to pocket the cash that the customer gave them, pretending that the sale did not happen. When the resulting discrepancy of inventory is eventually discovered, it could not be tracked back to the individual cashier in any way. To prevent this, retail stores have cash registers that automatically record sales transactions. Whenever appropriate, separation of duties should also be enforced in cloud computing. For example, separate technical user accounts should be used to create, archive, delete, and analyze log entries.

A final general principle is that roles (which specify a set of permissions) should not be assigned to individual users, but to groups. Individual users should then be assigned to the groups. While it is equally possible to assign exactly the same set of permissions to a number of users directly or with the use of intermediate groups, the latter is clearly preferable because it simplifies the management of permissions, and likely is less error prone due to the reduced complexity. In practice, this means that individuals that have the same job function and therefore require the same permissions should be placed in a group which is given the necessary roles (and consequently permissions) for this job function.

Google Cloud Identity and Access Management

On the Google Cloud Platform, access control is handled by Google Cloud Identity and Access Management (IAM). Its main functionality can be broken down into three parts:

1. Who ...
2. Can do what ...
3. On which resources?

The first aspect, "who", can be either accounts held by people or technical accounts. Accounts held by people, in turn, can be either normal google accounts, google groups accounts, accounts managed within a G Suite domain, or Cloud Identity accounts (i.e. accounts that are managed using Google's Identity as a Service (IDaaS) offering). Technical accounts used to give specific roles and permissions to google cloud resources are called service accounts. For example, when a new virtual machine is created in the GCP Compute Engine, by default it is run with the permissions of a default service account that is associated with the project that the virtual machine is attached to.

Default service accounts are very useful in the prototype phase of new applications, because they allow users to create resources without having to worry about assigning fine-tuned access permissions. However, the use of default service accounts is not recommended in a production environment, as it would likely violate the principle of least privilege. Instead, custom service accounts which grant exactly the required permissions (but nothing else) should be used.

Before we discuss the "can do what" aspect, it is easier to first look at the "on which resources" part. As explained earlier, resources on the Google Cloud Platform are organized in a hierarchy. At the top of the hierarchy is an (optional) organization node, which in turn contains (again optionally) a hierarchy of folders. Folders eventually contain projects, which in turn contain resources. Only the last two elements - projects and resources - are required: Any resource in GCP must be contained in a project, which, among other things, is used for billing purposes. Coming back to the "on which resources" question, permissions can be granted on individual resources, at the level of projects (in which case the permissions apply to all resources in the project), or at the level of folders or the whole organization, in which case once again the permission apply to everything contained in the folder or organization. Permissions are inherited from higher levels - in other words, the permissions an account has on a resource are the union of the permission this account has been granted on the resource itself, the project containing the resource, any folders that contain the project, and the organization node (if any) containing the folders.

The final part of the puzzle, the "can do what part", is defined by roles, which contain the actual permissions that users receive. There are three types of roles in GCP IAM: Basic (formerly called primitive), predefined and custom roles.

There are three basic roles - viewer, editor and owner. The viewer role allows read-only access to existing resources or data. The editor role contains the permissions of the viewer role and additionally allows actions that modify state, such as changing existing resources. Finally, the Owner role contains the permissions of the editor role and additionally allows managing roles, permissions and billing. The three basic roles can be applied at the level of projects or individual resources, and are basically a very coarsegrained legacy mechanism to manage permissions.

Predefined roles offer much more fine-grained management of permissions. They offer very specific permissions on different types of GCP resources. For example, the role storage.admin grants full control of buckets and their content, while the role storage.objectAdmin gives full control of objects contained in buckets (but not the buckets themselves). Predefined roles are designed to contain everything a specific job function (or technical functionality) typically requires.

Custom roles allow users to custom-tailor the exact set of permissions they want to grant to a specific role. In situations where the available predefined roles offer too many permissions, custom roles are necessary to fully apply the principle of least privilege. While they offer full control, custom roles have their disadvantages. In particular, since custom roles are completely managed by the user, great care must be taken to not accidentally grant too many or insufficient permissions in them. Otherwise, the principle of least privilege is violated, or applications might not function properly due to insufficient permissions. An additional argument for the use of predefined roles is that since they are managed by Google, they are also updated in the face of new GCP features. If custom roles are used, the roles have to be manually updated to allow their users access to new functionality. Therefore, in most instances, predefined roles should probably be used, and custom roles should only be considered if the existing predefined roles match up poorly with requirements.

6.3.3 Encryption

Encryption is a useful mechanism to secure IT systems in general. The facts that cloud computing is generally accessed over the Internet, and that public cloud computing features multi-tenancy, makes encryption particularly relevant in cloud computing

General Encryption Principles

Data in the cloud should be encrypted, both data “in transit” (i.e. data that is transferred over a network) and data “at rest” (i.e. data that is stored in secondary storage). The need to encrypt data that is being transferred does not need a lot of justification - since cloud computing is generally accessed over the Internet, it would otherwise be a relatively easy target for attackers. It might be less obvious that data stored on hard disks or other secondary storage devices should also be encrypted. There are multiple reasons for this requirement. Firstly, it protects data from unauthorized access if an attacker gains access to a server that can access the storage (e.g. due to a failure of virtual machine isolation). Secondly, it provides an additional layer of protection if an attacker manages to get physical access to the hard drive (e.g. by gaining unauthorized access to a data center). Finally, it reduces the risk that data is accidentally disclosed when secondary storage devices are discarded, e.g. because they failed or are replaced by newer models. Encrypted data on a hard drive is inaccessible if the encryption key is not available. This actually provides a very quick and convenient way to erase data - if the decryption key is destroyed, the data is effectively inaccessible, just as if it was safely erased[10].

For additional security in situations with particularly sensitive data, cloud service providers are starting to offer encryption not only of data in transit and at rest, but also while it is being processed. For example, the Google Cloud Platform offers so-called Confidential Virtual Machines, which use specialized features of current server CPUs to use in-memory encryption with encryption keys generated and stored by specialized hardware within the CPU. This provides a significant additional level of protection against attacks that manage to overcome VM Isolation, effectively eliminating risks caused by multi tenancy.

Encryption in the Google Cloud Platform

This section focuses on the encryption of data in rest. The encryption of data in transit is briefly covered in the section on network security below.

By default, the Google Cloud Platform encrypts all data that it stores, without any user interaction required. The mechanism encrypts chunks of data with so-called Data Encryption Keys (DEK), and encrypts these DEKs with so-called Key Encryption Keys (KEK). Keys are regularly changed (typically approximately every 90 days), and stored redundantly. The final point is very important: Since strong encryption methods are used, if the encryption keys were lost, so would be the data.

GCP users have three choices with regard to encryption: They can either rely on this default encryption mechanism, they can choose to use customer managed encryption keys, or they can use customer supplied encryption keys instead.

[10]Note that for extra security in case of particularly sensitive data, the deletion of a decryption key might not be considered sufficient by regulatory requirements, because the encryption could (at least theoretically) be cracked. In such cases, it is not uncommon to additionally physically destroy hard disks in an industrial shredder.

In the customer managed encryption approach, users still let GCP handle key creation, storage etc., but they take control of aspects such as the key rotation schedule. In other words, they control how often and when keys are changed. Using customer supplied encryption keys goes one step further - here encryption keys are provided by the user. This puts full responsibility for the creation, redundant storage, life cycle management etc. of the encryption keys into the hands of the user. This means that a potential attacker would have to gain access to both the encrypted data (in GCP) and the keys (managed by the customer), which can increase security if done well, and might be a necessary approach in areas with particularly strong regulatory requirements. The big disadvantage is that the organization has to take great care to not lose its encryption keys e.g. due storage media failures or ransomware attacks, because they would then also lose their data stored in the Google Cloud Platform.

6.3.4 Network Security

The importance of network security in a cloud computing should be self-explanatory given that cloud computing is accessed through computer networks.

General Network Security Principles
General network security principles in a cloud setting are essentially the same as in a non-cloud setting. Just like with access control, the principle of least privilege should be observed. In particular, firewalls should be configured in such a way that only the exactly needed traffic is allowed, but any other traffic (e.g. to ports or from sources or to destinations that are not necessary) should be blocked. This reduces the attack surface. Furthermore, all network communication should be encrypted. Encryption can happen at different levels, such as at the internet layer or application layer of the TCP/IP model with the use of IPsec or https, respectively. Depending on security requirements, it might be advisable to encrypt traffic independently at multiple layers. Finally, since attacks are often not immediately obvious, network traffic should be monitored carefully, for example by intrusion detection and/or intrusion prevention systems.

In cloud computing, some of the traditional pre-cloud network security approaches can no longer be applied. In particular, with in-house only IT, it is customary to partition computer networks into different segments, and to trust all requests that come from a “secure” segment. A common design uses three areas, the internal (safe) network, a so-called DMZ hosting servers that need to be accessible from both the internal network and the Internet, and the Internet (see Figure 6.2, source: the Search Securit website [11]). Between the internal network and the DMZ, and between the DMZ and the Internet firewalls are placed that only let authorized traffic through. In particular, the DMZ can be accessed from both directions, while the internal network cannot be reached from the outside (unless in response to a prior outgoing request).

If public cloud computing is used, no strict network separation between the different segments exists - machines in all three areas can be in the cloud. Consequently, a zero trust security model[12] is typically much more appropriate, where requests are no longer categorized as "safe" simply because they come from a trusted network, but where all requests are by default not trusted, unless they can be shown to be safe (e.g. by appropriate authentication and authorization mechanisms).

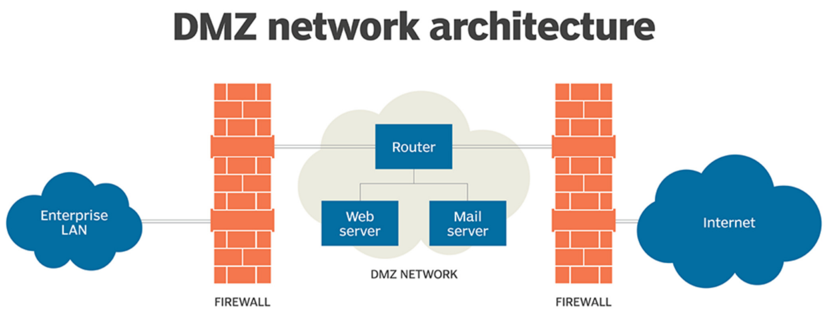

Figure 6.2: A network DMZ sits between two firewalls, creating a semi-safe buffer zone between the Internet and the enterprise LAN.

Network Security in the Google Cloud Platform

Resources in the Google Cloud platform such as virtual machines or Kubernetes clusters are by default connected through a Virtual Private Cloud (VPC) network. VPCs are global in scale, i.e. they can contain resources in any GCP data center, irrespective of its location, and consist of a number of regional virtual subnets. From the perspective of cloud resources, a VPN behaves like an exclusive physical network, even though it of course is routed through shared network connections. However, VPCs of different customers are strictly isolated from each other.

[11]`https://searchsecurity.techtarget.com/definition/DMZ`
[12]https://en.wikipedia.org/wiki/Zero_trust_security_model

VPCs are secured by a distributed firewall. While the firewall rules are defined at the network level, connections are denied or allowed on an instance level. What this means is that just like users do not have to manually configure (virtual or physical) switches and routers to connect their GCP resources, they also do not have to worry about placing (virtual or physical) firewalls at the appropriate locations in and between networks and subnets. Instead, rules that specify how network connections are to be treated at any point in the network are defined globally.

Firewall rules can be applied globally across the whole network, or limited to apply to tags that are attached to specific target resources. For example, a rule could specify that connections to port 443 are allowed on all resources (e.g. virtual machines) with the tag "webserver". The firewall is stateful. In other words, if for instance outbound traffic is allowed, return traffic (within a timeout window of ten minutes) is also allowed.

Firewall rules have the following parameters:

- **Target:** Which instances does the firewall rule apply to - this can be either specified via tags as described above, or via service accounts.
- **Direction:** Ingress or egress (i.e. inbound or outbound traffic).
- **Source (for ingress) or destination (for egress):** Individual IP addresses or address ranges for which the rule applies.
- **Protocol** (e.g. TCP, UDP or ICMP) **and port.**
- **Action to be taken**, either allow or deny.
- **Priority of the rule:** an integer between 0 and 65,535, with 0 being the highest priority, and 65,535 the lowest. Firewall rules are applied in order of priority, and the highest priority rule that matches a specific situation decides the action taken.

There are two implicit default rules with the lowest priority, allowing all outgoing traffic, and denying all inbound traffic. Additional default rules with the second lowest priority allow all VPC internal inbound traffic, and by default allow all ssh, rdp and icmp traffic.

Obviously, these default rules should generally be changed or overridden by additional (more detailed) rules with higher priority. The principle of least privilege dictates that only required network traffic is allowed. This is clearly violated by a default rule allowing all outbound traffic. In practice, only the ports that are actually used should be opened. This for instance makes it harder for an attacker who takes over a server to transfer data out of the network. It is also generally considered good practice to have rules with low priority that block all network traffic (in-and outbound) that is not explicitly allowed by rules with a higher priority. This way, a configuration error leads to traffic by default being blocked rather than allowed, which is better from a security perspective.

Furthermore, it is generally recommended to specify the target of rules via service accounts rather than via tags. The reason is that fewer permissions are necessary to attach tags to servers than to change the service account they run with. In other words, users with relatively limited permissions (such as administrative rights on individual virtual machines) could either inadvertently or with malicious intent attach tags to their virtual machine that allow more network traffic than should be permitted.

Also, since real-world firewall configuration can get complicated quickly do to the necessary number of rules, a good naming convention in which the name of firewall rules clearly reflects their meaning is also strongly recommended, to avoid potentially security relevant configuration errors due to misunderstandings.

Finally, in many cases (e.g. in hybrid cloud scenarios) it might be necessary to connect an on-premise network to a VPC. This can be achieved in different ways. If there are no particularly high performance requirements, a simple VPN using the IPsec protocol can be used. Cloud Interconnect is a different approach to connect directly (rather than through the public Internet) to the VPC using either a dedicated interconnect or a partner provider. These kinds of connections generally offer lower latency and higher reliability and scalability, but they are also significantly more expensive than a VPN connection.

6.3.5 Security Monitoring and Logging

The final area discussed in this chapter is the role monitoring and logging play in cloud security.

Monitoring and Logging for IT Security

Monitoring and logging are obviously done for reasons other than security as well, such as to keep track of resource use to ensure low response times and to keep costs under control. One of the main reasons why monitoring and logging are crucial for cloud security is that the occurrence of an attack is often not obvious at all - routine analysis of monitoring data or logs might in many cases be the best way to detect an ongoing attack and to prevent and minimize resulting damage. Similarly, once an attack has been detected, monitoring and logging data can be very helpful to determine the scope of the attack and to figure out which weaknesses allowed the attack to succeed in the first place, and consequently should be mitigated to prevent future attacks from succeeding.

Monitoring and Logging in the Google Cloud Platform

The Google Cloud Platform offers comprehensive monitoring and logging features. The monitoring functionality includes collecting a wide range of metrics, such as uptime and response times of services, utilization rates of resources such as CPUs, networks etc., many of which can be useful indicators to detect for instance denial-of-service attacks or other factors that influence for example the availability of services. Monitoring data can be visualized in custom dashboards that make it easy to compare current to historic data, and it can be used in alerting policies that trigger notifications if certain conditions occur or are violated (e.g., if a service does not respond). While they are not conclusive, significant deviations of monitored metrics from normal values can often be a sign of an attack.

The logging functionality allows centralized access to and management of logs that would otherwise be spread out through the whole application and all its resources. Of particular interest from a security perspective are admin activity logs and data access logs. The former logs creation, deletion and other changes of resources. Admin activity logs cannot be disabled and are by default kept for 400 days. Data access logs keep track of which users access data in GCP. Depending on the applications and its data use, these logs can get very large, which is why they are turned off by default, and why the default storage duration if turned on is only 30 days. Still, especially if data is sensitive, data access logs can play a key role in keeping data secure.

Because logs can grow very large, a life cycle should be defined for them, i.e. there should be clear rules about how long logs are kept before they are deleted, or at least archived (e.g. for auditing or compliance reasons) to cheaper, longer term storage such as coldline storage buckets. The size of log files is not only a potential issue with regard to storage costs - it can also make it hard to extract useful information for security purposes from them - the analogy of searching for a needle in a haystack comes to mind. To make this task easier, log management in GCP offers powerful features to aggregate, filter or even statistically sample log data from multiple resources or even projects, and to write the resulting consolidated log data to cloud storage (e.g. as simple text files) for long time archiving, to BigQuery to allow for SQL queries against the log data to analyze it in more detail, or to PubSub to export it to other tools.

6.4 Summary

In summary, securing applications in the cloud is a complex undertaking that can be achieved by no single product or technology. Instead, a defense-in-depth approach should be used, in which multiple lines of defense are used to make it as hard as possible for attackers, and to increase the chance of the discovery of attacks before they achieve their ultimate targets. The additional risks incurred by using (especially public) cloud providers can, if done correctly, be overcome by taking advantage of the increased level of security cloud service providers can offer in their area of responsibility due to their size and experience. However, it is critical to realize that the security of the cloud (i.e the infrastructure operated by the cloud provider) needs to be complemented by the security in the cloud (i.e. of the applications deployed in the cloud) to achieve an acceptable level of overall security.

Chapter 7

Multi-Cloud Systems

As discussed in this[1] paper, in the last few years, cloud computing has become a very popular and effective solution for enterprises to provide their services on a "pay-per-use" basis. In particular, with the Infrastructure-as-a-Service model, users can rent virtualized resources, hosted by remote data centers, where they run their services inside isolated execution environments like Virtual Machine instances and containers.

Cloud companies such as Amazon, Google and Microsoft have developed their own platforms featuring proprietary web, command-line, and programming interfaces, by means of which customers can manage their VMs and analyze the performance of their applications.

Unfortunately, the lack of standard interfaces to access these platforms makes it difficult for customers to switch between different providers. This can limit their cloud experience in many ways. For example, it makes it hard to

- figure out which one of the cloud platforms provides the best solution that fits with their needs or to combine services of multiple providers to get the best from each of them.
- use private and public cloud together (e.g. to use already existing in-house hardware, to be not fully dependent on external providers etc.)
- use multiple cloud providers to avoid being fully dependent on a single cloud provider in case of contractual disputes, the provider going

[1]Cosimo Anglano, Massimo Canonico and Marco Guazzone. EasyCloud: Multi-clouds made easy. IEEE Computers, Software, and Applications Conference (COMPSAC), 2021.

out of business, increasing prices, the discontinuation of a specific product/service used etc.

- realize the extra reliability of using multiple cloud providers (e.g. in case of a single provider screwing up their internal provisioning, networking, authentication or other systems).

This situation often leads to the so-called vendor lock-in, whereby users are effectively locked into specific providers, and where new cloud providers cannot easily enter the market.

To overcome these limitations, Multi-cloud Systems (MS)[2], i.e. cloud infrastructures composed of resources drawn from different cloud platforms, are increasingly often used as a way to combine services or resources of different cloud platforms by means of a unified interface. That way, customer organizations can exploit the advantages of different cloud providers (e.g., in terms of service cost, quality of service, and performance) without being locked into a single one. To create a MS, a suitable toolkit, providing the necessary "glue" between different cloud infrastructures and the appropriate level of abstraction, is required. In particular, such a toolkit should provide the following features:

- **Interoperability**, i.e. the ability to support multiple cloud platforms, so as to avoid vendor lock-in;

- **Platform independence**, i.e. the provision of a unified, platform-agnostic user interface that hides the API heterogeneity of these platforms;

- **Effective resource provisioning**, i.e. the ability to ensure that the VMs delivering a given cloud service will be sufficiently resourced in a timely manner as load increases, so as to achieve desired levels of performance, efficiency, and availability;

- **Ease of use**, i.e. the provision of suitable mechanisms enabling users to interact with ease and effectiveness with the MS.

Various toolkits have been developed from both academia and commercial providers to allow smooth cloud interoperability and to harness multi-cloud heterogeneous resources. Unfortunately, they provide only a subset of the features mentioned above, so we can assure that an efficient

[2]A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected cloud computing environments: Challenges, taxonomy, and survey,"ACMComput. Surv., vol. 47, no. 1, 2014

and complete MS is still missing at the time of writing. The rest of this section briefly discusses the main characteristics of some of the most important MS toolkits.

7.1 MS from Academia

- **FSToolkit**[3] enables inter-cloud federation: the user provides credentials for the cloud platforms that they want to interact with, and can then create one or more federation scenarios. In a federation scenario the user specifies which resources and services, belonging to different cloud platforms, they want to use in their experiment. FSToolkit provides a graphical user interface and has a default plugin able to interact with AWS, while plugins for different cloud platforms can be created. The main problem of FSToolkit is its obsolescence, since it appears to not have been updated since October 2012. This means that FSToolkit is practically unable to deal with present cloud platforms, as typically cloud providers update the internal code of their platforms every few months.

- **Apache JClouds**[4] is an open source multi-cloud toolkit for the Java platform that facilitates developing applications for a wide range of cloud platforms. JClouds supports about 30 cloud providers and cloud software stacks, including AmazonAWS, Microsoft Azure, OpenStack and Google Cloud, just to name a few. Moreover, it offers several API abstractions as Java And Clojure libraries. Despite its many features, JClouds does not provide any effective resource provisioning mechanisms (for instance, it is not possible to check the health of the VMs), and provides ease of use to a limited extent, since it requires Java programming skills to be used.

- **Apache Libcloud**[5] is an MS toolkit, provided in the form of a library, written in Python and allowing interactions with several popular cloud service providers. In particular, Libcloud provides a unified API able to hide the differences between the APIs of different cloud providers. As JClouds, Libcloud does not provide any effective resource provisioning mechanisms (monitoring and performance mea-

[3]http://nam.ece.upatras.gr/index.php?q=node/35

[4]E. Toews and D. Advocate, "Introduction to apache jclouds," Apr, vol. 7,p. 23, 2014.

[5]Apache Software Foundation. (2021, Jan.) Libcloud. [Online]. Available:https://libcloud.apache.org

surements are not provided), and provides ease of use to a limited extent, since it requires Python programming skills to be used.

- **Cloudmesh**[6] is another MS toolkit able to provide access to various cloud platforms such as AWS, Azure, Google Cloud And OpenStack. It has a variety of repositories that add features to Cloudmesh based on needs by the user. Compared to JClouds and Libcloud, Cloudmesh is easier to use, but it also does not provide any effective resource provisioning mechanism.

- **Occopus**[7] is a MS toolkit supporting various cloud platforms that is focused on resource orchestration and management. In particular, it allows describing virtual infrastructures and node definitions in a cloud agnostic way, and can automatically deploy and maintain the specified virtual infrastructures in the target clouds. Occopus provides only some of the mechanisms needed to achieve desired levels of performance, efficiency and availability (for example, it provides the health monitoring of the VMs). In particular, Occopus needs to exploit external tools like Prometheus to provide auto-scaling mechanisms. For this reason, Occopus provides only partially effective resource provisioning features.

- **EasyCloud**[8] is a MS toolkit able to effectively support the creation and use of MSs by providing interoperability and platform-independence by means of an extensible cloud interfacing subsystem. EasyCloud also provides effective resource provisioning by coupling two separate mechanisms, namely:

  - VM metrics collection and dispatching that collects in (near) real-time user-defined metrics data (e.g., CPU and memory load) from VMs and dispatches them to multiple "sinks" for storage as well as for further analysis and processing, thus enabling users to analyze and understand how their applications and services are performing;

[6]G. Von Laszewski, F. Wang, H. Lee, H. Chen, and G. C. Fox, "Accessing multiple clouds with cloudmesh," in Proceedings of the 2014 ACM international workshop on Software-defined ecosystems, 2014, pp. 21–28.

[7]J. Kovács and P. Kacsuk, "Occopus: a multi-cloud orchestrator to deploy and manage complex scientific infrastructures," Journal of Grid Computing, vol. 16, no. 1, pp. 19–37, 2018.

[8]C. Anglano, M. Canonico and M. Guazzone. An educational toolkit for teaching cloud computing. In ACM SIGCOMM Computer Communication Review, 51(4), 2021.

  – VM monitoring and provisioning that exploits collected metric data to monitor the performance of VMs in realtime and to trigger management actions according to user-defined policies (e.g., to implement load balancing and autoscaling).

Finally, EasyCloud provides ease of use by means of a unifiedAPI that frees users from learning the different proprietaryAPIs exposed by the various cloud platforms it supports, and by an interactive and intuitive user interface that allows even inexperienced users to easily manage their VMs. The main problem of EasyCloud is similar to other MS: delayed updates of the tool in response to modifications of the API of the supported cloud platforms. EasyCloud is managed by faculty staff members (mainly assistant professors and associate professors) that are not always involved in the software development in their duties. This means that when, for example, OpenStack decided to change the API of the monitoring component, EasyCloud took almost three months to update its code in order to work with the new component of OpenStack.

7.2 Commercial MS

In this section we propose a subset of the most relevant commercial multi-cloud systems available at the time of writing. Due to the dynamicity of the cloud computing, the list of most relevant products could change very quickly. From our point of view, it is still important to present what these solutions claim to provide to their users. Most of them proposes a free trial period so the interested reader can evaluate the actual features provided by each system.

- **CloudHealth**[9] is a product provided by the company VMWare which claims to provide the following main features [10]: data collection, performance reporting and analysis, cost management (that is, compute which user/department is spending most) and user management (authorization and role-base access control).
- **Morpheus**[11] is a Linux package which claims to be able to manage both on-premises private clouds from existing hypervisors (like VMWare and Nautilus) and public clouds (like AWS, Azure, GCP, etc.).

[9]https://www.cloudhealthtech.com/
[10]https://searchcloudcomputing.techtarget.com/definition/CloudHealth-Technologies
[11]https://morpheusdata.com/

- **Cisco CloudCenter Suite**[12] claims to provide two key features besides all the common feature to interact with different cloud platforms: cost optimization (in charge to attempt to reduce the cloud spend through optimization recommendations) and DevOps integration (in charge of trying to reduce human error by automating workflows and repetitive tasks).

- **OpenShift**[13] is developed by Red Hat and it claims to be designed to allow applications and data centers to scale up/down quickly by exploiting containers both in on-premises and multi-cloud environments.

Finally, there are tools that cannot be considered as MS but they can help with integrating multi-cloud systems:

- **Terraform**[14] is an open source tool which claims to be focused on the software delivery process through code instead of interacting with console or command line interfaces (this approach is called *Infrastructure as Code* (IaC)).

- **TOSCA**[15] is an acronym for Topology and Orchestration Specification for Cloud Applications and it proposes an XML-based modeling language to address three problems: (1) automated application deployment and management, (2) portability of applications and their management, and (3) interoperability and reusability of components.

The projects mentioned in this section are just a subset of all multi-cloud solutions provided by the scientific community and by companies.

7.3 Summary

In this chapter we discuss various multi-cloud systems by explaining pros and cons for each of them. Due to the dynamicity of the different cloud platforms these considerations could become obsolete very quickly, but it

[12]https://www.cisco.com/c/en/us/products/cloud-systems-management/cloudcenter-suite/index.html
[13]https://www.openshift.com/
[14]https://www.terraform.io
[15]Binz et al., "TOSCA: Portable Automated Deployment and Management of Cloud Applications", Advanced Web Services, 2014

is worth mentioning that the lack of standardization in the way cloud services are operating, portability of cloud services and cloud security can potentially slow down the innovation process in cloud environments. Finally, as mentioned before, we report what the various multi-cloud systems claim in their documentation: the actual pros and cons of the solutions proposed must be tested. To the best of our knowledge, an objective academic paper comparing the various multi-cloud systems is still missing.

Chapter 8

Cloud Operations

The previous chapters are mostly concerned with technical aspects of and design decisions related to the use of cloud computing - including topics such as the best choice of deployment and service models, different architectures for cloud applications and how to design secure systems in the cloud. Once all these decisions have been made, another important area is to make sure that cloud applications can be operated efficiently and effectively.

Operations is of course not a completely separate issue. For example, when the security measures of an application are designed, the way it will be operated has to be considered carefully, including how users and their access to the application are managed operationally. In general, as is discussed briefly in the chapter on Cloud Architecture, modern software engineering approaches usually contain the concept of DevOps, which strives to better integrate development and operations of an application.

Still, there are a few areas of cloud operations that deserve to be covered in more detail in this chapter. We begin by looking at a number of different areas of managing cloud operations, followed by an overview of cloud monitoring and migration. A brief review of challenges and limitations of cloud computing and of some potential ethical issues complete the chapter.

8.1 Cloud Management

One of the key reasons why managing an application in the cloud is different from managing a non-cloud application is that a number of responsibilities for the application are split between the cloud service provider and the cloud service user. While the details depend on multiple factors, including the cloud service and deployment models chosen, several areas require specific management attention irrespective of the choices in a given situation. They are described in this section.

8.1.1 Service Level Agreements (SLA)

One key management tool to handle split responsibilities between a service provider and a service user is the Service Level Agreement (SLA). SLAs specify a number of issues such as

- The percentage of uptime for a service (e.g. 99%, 99.9% or 99.99%)
- Guaranteed network bandwidth and performance and/or capacity of other resources
- The consequences if these guarantees are not met (typically (partial) refunds)
- Planned maintenance windows of resources
- Maximum help desk response times, escalation levels etc.

An SLA is typically a legally binding contract that allows the organization using cloud services to assess to what degree the guaranteed service levels are sufficient for them. For example, while a 99.9% uptime might sound like a lot, the allowed downtime per year is still almost nine hours, which, if they were to occur at a bad time (e.g. for an online retailer on a major shopping day such as Black Friday) might still be very damaging. Additionally, if the guaranteed uptime was violated, and the contractual consequence (e.g. a partial refund of the monthly service fees) was triggered, the damage caused by the lost business due to the outage might significantly exceed the applicable refund. Consequently, based on such an SLA, an organization with high reliability needs would probably take extra steps to design its application more fault tolerant, e.g. by using cloud resources in multiple different data centers, possibly even by multiple cloud service providers, to achieve the required reliability.

SLAs are not only helpful to organizations using public cloud providers. Even in private cloud settings they are often used as formal contracts between different organizational units, e.g. the IT department operating the cloud resources, and different business units using them for their applications.

8.1.2 Organizational Measures

Just like software development differs significantly between e.g. a student writing a small program for a class assignment and a team of dozens of developers writing a large business application, managing cloud applications or cloud infrastructure for complex, real-world cloud deployments requires significantly more organizational measures than doing a small assignment in a cloud computing class in order to be successful. In a modern real-world cloud deployment, automation allows users to perform changes to large numbers of systems with often a single command. While this is generally a good thing - it allows a small number of administrators to manage a large cloud deployment - it also means that small errors can have large effects.

For instance, AWS experienced an outage on March 2, 2017 that affected significant parts of the Internet for several hours. This outage was the result of an error by an operator who tried to take a small number of servers offline for planned maintenance, but entered one of the inputs to the executed command incorrectly, resulting in a much larger number of servers being taken offline than intended[1]. While this is just one example of a small error resulting in significant damage at a public cloud provider (and its customers), it is clear that similar issues could happen at any organization operating or using cloud resources at a large scale. Consequently, the management of cloud resources, irrespective of whether the organization is a cloud service provider, user, or both, requires particularly careful organizational measures to reduce errors and their consequences as much as possible[2].

Two commonly used organizational measures are policies and procedures. Policies are rules that an organization sets itself, while procedures describe the steps that need to be taken to achieve certain tasks. In an organization that operates applications in a public cloud, a policy could specify who in the organization has the authority to approve different types of changes to applications. A related procedure implementing such a policy could list the specific steps that need to be taken, such as required sign-offs by development leads, test managers etc. Both policies and procedures standardize processes and can, if used appropriately, reduce errors and re-

[1]https://www.geekwire.com/2017/amazon-explains-massive-aws-outage-says-employee-error-took-servers-offline-promises-changes/

[2]Clearly, as a leading public cloud provider, AWS is aware of this, and is surely taking significant organizational measures. The fact that such problems can still occur further illustrates the need to build cloud applications with multiple levels of fault tolerance.

sulting problems.

This section barely scratches the surface of possible organizational measures. There are several frameworks related to IT operations in general that also apply in a cloud computing context, including ITIL[3] and COBIT[4].

8.1.3 Workload Management

One of the key advantages of cloud computing that is discussed in multiple contexts in this book is that in a cloud setting, applications can dynamically allocate and deallocate resources. In contrast to this, the traditional way to deploy applications on dedicated physical servers gives applications a fixed set of resources that can only be changed slowly by up- or downgrading the hardware that they are deployed on. For individual applications, this flexibility is great - if they are designed to scale automatically, there is no need to accurately predict usage patterns such as peak time loads to prevent overloaded or underutilized servers.

Of course the need to make sure that there are sufficient computing resources available at all times does not disappear in cloud computing - the responsibility is simply shifted to the cloud provider. The fact that cloud applications get resources from a pool of available virtual resources means that resources deallocated dynamically from one application become available for other applications. Therefore, cloud providers try to host a wide range of different applications with peaks and troughs that do not occur simultaneously, in order to maximize resource utilization and reduce the risk of resource shortages.

Another tool cloud providers use are discounted non-guaranteed instances. For example, AWS, Azure and Google offer so called Spot Instances[5] [6] [7]. Spot instances behave like normal virtual machine instances, except that they are significantly discounted (for example by 75%), and that Google reserves the right to shut them down on very short notice (just sufficient to have a proper shutdown of the deployed applications) if they need the resources for other users. In effect, spot instances serve as a buffer, significantly reducing the risk that the cloud provider runs out of instances when too many applications deployed on regular virtual machines attempt to allocate more resources at once.

[3]https://www.axelos.com/best-practice-solutions/itil

[4]https://www.isaca.org/resources/cobit

[5]https://aws.amazon.com/ec2/spot/

[6]https://docs.microsoft.com/en-us/azure/virtual-machines/spot-vms

[7]https://cloud.google.com/spot-vms

In addition to this, cloud providers need to carefully monitor the utilization of their resources, and model future resource needs based on past utilization patterns. Given the still strongly growing market for cloud computing services, the question is ultimately not if additional computing resources will be needed in the future, but at which rate they need to be added to keep up with demand. Still, if unexpected general peaks in demand occur, cloud providers can shut down spot instances, or even increase the discount given to spot instances to encourage even more users to use them and to consequently increase the available resource buffer.

The problem of ensuring sufficient capacity to satisfy occurring workloads is typically more difficult for organizations that use a private cloud, because the set of applications in the cloud is given by the IT needs of the organization, and could well be rather homogeneous with co-occurring peaks and troughs of demand. This, of course, is often a key motivation for the use of a hybrid rather than purely private cloud, so that additional public cloud resources can be used during peak workloads.

8.1.4 Mobile Device Management

Mobile Device Management might seem like an odd section in a book on cloud computing, where applications are deployed in large data centers and not on mobile devices. However, the shift to cloud computing has generally made it much easier to access applications from a wide range of devices, including devices with limited computational resources. In contrast, when applications were operated following more traditional client-server paradigms, they could only be used on devices with client-side software installed. Modern applications deployed on the cloud usually require only network connectivity and a web browser, requirements which are fulfilled by a wide range of devices including tablets and smartphones.

As already briefly discussed in the chapter on security, the fact that applications and their data reside in a centralized location in the cloud (and are not distributed across a large number of end user client devices) is a security opportunity of cloud computing. In contrast, in more distributed client-server architectures or completely client side applications, the loss or theft of a client device (such as a laptop or smartphone) potentially puts all the data of the application into the hands of the finder or thief. One of the chief purposes of mobile device management is to ensure that the cloud application and its data remain safe from attackers when mobile devices fall into the wrong hands.

In order to achieve this, mobile device management systems have a wide range of features. For example, they can typically enforce security settings on the devices. For example, if an employee wants to access a cloud application with sensitive data from their phone, the phone has to have at least a five digit PIN which has to be entered to unlock the screen. Additionally, the device has to lock after five unsuccessful attempts to enter the PIN, and the screen has to lock automatically after the device has not been used for more than a few seconds. Additional common features include the ability to manage phones remotely, including the ability to remotely lock or wipe phones that have been reported lost or stolen, to locate missing devices, or to have separate environments on the phone for company use and personal use (e.g. separate address books etc.). There are many other mobile device management features such as the ability to automatically manage VPN connections, to make sure that security updates are installed when available and to collect usage data and device diagnostics.

Properly used mobile device management can significantly increase the security of a cloud application while at the same time allowing users the flexibility to use their personal mobile devices to access company applications.

8.2 Cloud Monitoring

Any significant IT system or infrastructure is subject to a wide range of issues that can negatively affect its users, such as power or network outages, other physical hardware failures, software bugs, security breaches as a result of outside attacks etc. The main purpose of monitoring systems is to detect any such issues as early as possible, ideally before they impact end users negatively, and to support operators in fixing the underlying causes as quickly as possible. This is equally true for cloud systems as it is for non-cloud systems. The following two sections compare monitoring in the cloud to general monitoring, and illustrate some of the monitoring features offered by the Google Cloud Platform.

8.2.1 Introduction

There are two main aspects of how monitoring in cloud deployments differs from monitoring of non-cloud IT systems. The first one is that monitoring (just like many other aspects such as security) is a split responsibility between cloud service provider and cloud service user. In general, cloud service users are not responsible for and consequently do not have to monitor the underlying physical and network infrastructure. The exact boundaries of which parts are monitored by the provider as opposed to the cloud service user depend on the service model chosen. While there is some overlap, monitoring systems used by cloud service providers and cloud service users focus on different aspects. Cloud service providers are more interested in monitoring system usage levels carefully to support their workload and capacity planning as well as general data center management. Cloud service users care more about monitoring their application on higher levels of the cloud stack, often with application specific aspects such as login errors or end-user response times being of particular interest. Monitoring of some areas such as SLA compliance and billing would be equally relevant for both.

The second difference is that monitoring of (at least modern) cloud applications has a different focus. This can be best explained with the pet vs. cattle[8] paradigm shift in cloud computing, according to which traditional administrators manually managed (and consequently monitored) individual servers, while cloud deployments are made up of a large number of easily (often automatically) replaceable systems that are administered (and consequently monitored) as a collective. To illustrate this point, a traditional non-cloud web server might have been deployed on four physical servers, the health, load and performance of each a system administrator would have carefully monitored. In a modern cloud deployment, the same web server might run on an automatically scaling instance group of virtual machines or a Kubernetes cluster with multiple nodes. In such cloud deployments, the health of each individual virtual machine or node is less interesting, because they are automatically restarted or replaced in the event of failures. Instead, the focus of monitoring shifts more towards the application and deployment as a whole.

Cloud monitoring systems for cloud service users typically have a centralized user interface that displays data from all the cloud resources used. This data can either be collected by installing small programs called agents on all the systems that are monitored, or simply by having a centralized monitoring application automatically connect to the systems regularly e.g. via ssh and collect data about e.g. disk, memory or CPU usage using standard operating system commands. While agents increase the overhead a little because they first have to be installed, and because they take up some memory and CPU time on the monitored system itself, they tend to offer richer and more real-time data than agentless systems can.

[8]Tilkov, S. "The Modern Cloud-Based Platform", IEEE Software, March/April 2015

8.2.2 Monitoring in GCP

The monitoring component integrated in the Google Cloud Platform uses two separate agents to collect monitoring and logging data on monitored machines. For customer managed cloud resources like virtual machines, the cloud service user has to install these agents to enable detailed monitoring.

One of the key monitoring features in GCP are alerting policies. Alerting policies are based on conditions, which in turn define that certain metrics exceed a specified threshold for a defined minimum time period. For example, a metric could be the mean CPU utilization on a virtual machine, a threshold could be 80%, and the minimum duration could be one minute. In this case, the alerting policy would trigger if the CPUs of a virtual machine have a load of 80% or more for at least one minute. A very wide range of metrics can be used, covering all kinds of cloud resources. For example, virtual machines or containers can be monitored just as well as cloud databases or flat cloud storage, or network traffic and service and other user accounts. A special type of condition is an uptime check, which simply regularly attempts to connect to a server (e.g. by sending an HTTP(S) request in the case of a web server) from multiple different locations in the world, and records responses and response times.

Once an alerting policy is triggered, an incident is raised. Initially the incident is in the state "open". An administrator can change the state to "acknowledged" to indicate that the incident is under investigation, and finally to "closed" once it is resolved. Incidents can also be configured to trigger notifications through a number of different notification channels such as emails or text messages. This is very useful especially for smaller organizations which may not have system administrators monitoring their cloud applications 24/7, but that can now put an administrator on call to be alerted automatically if serious problems occur. Another important monitoring feature are dashboards. Dashboards allow administrators to combine all the important monitoring information of a number of systems in one display. GCP comes with a number of default dashboards that provide generally useful summary information about different types of cloud resources such as storage, firewalls or VM instances. Users can create their own custom dashboards to display whatever data is most important and organize them in the way that serves their individual needs best. On dashboards, data is often represented in graphs, because humans tend to be much quicker at gaining useful information from data that is represented graphically as opposed to in textual form. The graphs often show how important metrics change over time, which is another important feature, because for many metrics, an absolute number is hard to interpret. For example, if the learning management system of a mid-size college has forty failed login attempts in an hour, is that a potential sign of an attack, or simply the normal volume of students forgetting their passwords? Being able to (visually) compare current with historic data (e.g., how many failed login attempts did we have in the same hour yesterday, or last week, or last term?) makes it much easier to interpret.

Finally, GCP cloud monitoring allows easy and centralized access to log data. Real-world cloud applications are often deployed on a significant number of systems (e.g. virtual machines), and might consist of different types of components for different purposes - for instance, web and application servers, relational databases, non-relational databases, connectors to messaging systems etc. All these underlying machines, and all the different framework elements, applications etc. tend to write their own separate log files. Manually retrieving and analyzing the logs on multiple machines, for example to analyze a bug report by end users, would be a rather labor intensive task. Therefore, the GCP logging agent collects all the relevant log files, and the logging component provides users with a centralized logs explorer. In the log explorer, queries can be defined to retrieve information of interest from the logs. Even here, graphical representations are used to help users make sense of the data. For example, users can generate histograms of log files which show how the number and relative frequency of log entries of the different severities (e.g. warning, error, critical) changed over time.

8.3 Cloud Migration Strategies

When organizations adopt cloud computing, they can either do that for newly created applications, or they can move applications that were previously operated on in-house servers. It certainly happens that completely new applications are created in the cloud - due to the minimal up-front costs for physical hardware and infrastructure and the inherent elasticity of especially public cloud providers, this is certainly an attractive option for startup companies, which tend to have completely new applications. However, the majority of new cloud deployments are likely of previously existing non-cloud applications. This raises a number of questions about how to perform such migrations into the cloud.

8.3.1 Perspectives to Consider

When an organization decides to migrate an application into the cloud, there are a wide range of perspectives to consider. The most obvious one is the technical perspective, which primarily focuses on how the application can be implemented on the cloud computing platform. This includes aspects such as which compute service to use (e.g. virtual machines, containers, or some PaaS offering), how data should be stored (in a relational database, in a noSQL database...) etc.

Another important perspective is operations. Getting the application to run in the cloud is one thing, but it also has to be operated, which includes a wide range of ongoing routine tasks such as end user support, bug fixing, the installation of updates, monitoring, backups etc. A lot of these tasks change significantly when an application is moved from an in-house deployment to a cloud service provider. For example, some tasks that previously used to be completely under the control of in-house IT become split responsibilities between the cloud service provider and the organization using cloud services: Depending on the Service Model chosen, e.g. different parts of monitoring and of the installation of updates become the responsibilities of the provider. While the cloud provider is only in charge of the physical hardware and the networking infrastructure in Infrastructure as a Service, it takes on additional responsibilities for installed operating systems and other platform elements if Platform or Software as a Service is used. Similarly, while cloud users likely retain some responsibilities for backups, the actual process of scheduling and performing backup and restore operations is likely quite different in a cloud deployment. Consequently, existing operations handbooks, policies and procedures need to be completely rewritten when an application is migrated into the cloud.

Security is another key aspect. It affects both the implementation of the application in the cloud and its operations. Once again, compared to an application that is hosted in-house, parts of the security responsibilities such as physical security of the underlying hardware are taken on by the cloud provider. Different components of a security architecture likely need to be adjusted when an application is migrated to the cloud, for example firewall rules and other parts of network security or user authentication mechanisms. Another key consideration should be the impact on people in the organization. Some tasks such as taking care of the physical IT infrastructure will either be eliminated or significantly reduced by the migration. This can lead to justified fears about job security, and result in resistance to the changes that come with a migration into the cloud. Even employees whose jobs are not at risk might see their responsibilities significantly changed, and likely will have to learn a number of new skills to still be successful in their roles. This is likely true for most IT personnel, ranging from developers and architects over testers to operators, all of whom need to be trained in cloud technologies.

All the perspectives discussed so far are directly related to technology. However, other business functions and the business overall also need to be considered. For example, Human Resources departments likely have extra work to change the roles of employees and to possibly hire and lay off other employees. Similarly, Finance and Accounting departments have to adjust to much more volatile costs (as cloud applications scale up and down based on load), and different levels of management have to deal with significant migration projects, often with the temporary addition of external consultants and other service providers. In order to achieve the full benefits of a migration to the cloud, even departments that are not directly affected need to embrace the changes that come with cloud computing. For example, in order to take full advantage of the increased technical agility that comes with cloud computing, such as the ability to scale, change and update applications more rapidly than before, a higher level of organizational agility is necessary, so that new capabilities are actively exploited when business opportunities to do so arise.

In summary, in order to successfully migrate (parts of) an organization's IT into the cloud, significantly more aspects than just the technical implementation need to be considered.

8.3.2 Migration Assessments and Prioritization

When an organization begins considering moving parts of its IT into the cloud, a number of detailed assessments should be performed to determine whether a migration to the cloud is in fact beneficial overall, and for which specific applications in particular. These assessments partially relate to key factors that determine whether there is a good business case for migration into the cloud, including:

- Reduced costs, both bound capital in the form of physical IT resources, as well as ongoing costs in the form of rent, electricity, payroll etc. to operate IT resources

- Operational resilience, a higher level of which can typically be achieved more easily and more cost effectively with a cloud provider, due to the wide geographic distribution of available data centers (resulting in reduced vulnerability to local outages caused e.g. by natural disasters), the practically unlimited scalability offered by cloud providers etc.

- Increased agility and productivity due to the ability to rapidly scale cloud resources up and down, to provision additional test environments, to roll out new versions of applications etc.

A comprehensive business case also has to include costs and other disadvantages of a cloud migration. An example of an (at least temporary) negative effect of a cloud migration is the cost incurred by the migration itself, which, considering the wide range of perspectives that have to be appropriately considered as discussed in the previous section, typically is a sizable change project with corresponding costs.

The following are some of the key assessments that should be performed:

- **Financial Assessment:** A detailed financial assessment is often done in the form of a Total Cost of Ownership (TCO) analysis, which compares all costs over the whole (typically multi-year or even multi-decade) lifecycle of an application, both in a cloud and a non-cloud scenario. Such an analysis has to include factors like the cost to maintain or retire existing IT resources such as servers and networking equipment, but also software licenses and long-term support contracts, as well as short-term costs for potential cloud migration projects. Such a detailed analysis heavily depends on assumptions about factors such as the future growth and volatility of the processing load of the application. For example, with a relatively stable and predictable load, in-house IT resources can achieve consistently high utilization and consequently be fairly cost effective. In contrast, if the load fluctuates strongly, in-house IT resources would have a low average utilization rate and therefore not be cost competitive compared to a public cloud in which only actually used resources have to be paid for.

- **Security and Compliance Assessment:** Another important assessment revolves around the security and compliance requirements of the application. As discussed in the chapter on cloud security, the use of cloud computing brings both cloud security risks and opportunities, the importance of each of which has to be analyzed in the given situation. Typical factors to consider are how sensitive the data that an application uses is, what security threats the application might face, and to what degree these threats can be mitigated by security measures offered by different cloud providers. In addition to these considerations, organizations in many industries have to comply with government imposed security and privacy regulations. This can especially be problematic when data crosses regulatory boundaries. To just give a simple example, educational institutions in countries such as Canada have to follow strict standards to protect personal data about their students. This often prevents them from storing such data in public cloud data centers in countries like the USA which have lower privacy standards. Another example is that financial institutions are typically required to maintain specific certifications in order to reduce the risk of failures of their IT systems, because such failures would have wide-ranging impacts on many other companies. Such requirements do not necessarily prevent the use of public cloud computing. However, they typically require that a public cloud provider which itself carries the required certifications has to be chosen. For example, the Google Cloud Platform supports a number of compliance requirements from a wide range of industries in all major regions of the world[9].

- **Technical Assessment:** The third kind of assessment revolves around technical aspects of the application. A critical issue in this area especially for legacy systems is whether cloud platforms under consideration support the platforms, protocols and tools required by the application. Older, mainframe based legacy applications might require operating systems or networking protocols that are not directly supported by the virtualization layer of the cloud provider. Similarly, an organization that uses old, uncommon or custom development tools such as version and configuration management tools might find it challenging to integrate them with a modern public cloud provider. Another aspect are performance requirements of the application. For instance, applications that transfer significant amounts of data or require very short response times might find network bandwidth or latency limiting factors that could make the use of public cloud providers either impractical or uneconomical.

These assessments are quite interdependent- for example, security requirements or technical challenges can result in (or be resolved by) additional measures that cause additional costs, and thereby affect the financial assessment. It should also be noted that the assessments typically do not

[9]Compliance information for the GCP can be found at https://cloud.google.com/security/compliance

result in a binary yes-or-no answer. While it might occasionally happen that a specific assessment shows that the migration to the cloud is infeasible for a specific application, the more likely outcome is a possible, but more or less challenging migration.

In practice, most established organizations have significantly more existing applications than they can realistically migrate into the cloud at once, for example because of limited personnel and financial resources. In such a situation, the various assessments can form a solid basis on which prioritization decisions can be made. The ultimate decision on which applications should be migrated to the cloud first often takes additional factors into account. While it might be tempting to begin with whatever application would benefit most from a move into the cloud, for example because it is highly volatile and could consequently lead to significant cost savings, many organizations also consider the relative importance of the different candidates for the organizations. Especially if there is little in-house expertise with cloud computing and migration projects, it might be best to begin with less important applications (even if they do not promise the highest migration return), simply to reduce risk by building expertise when less critical applications are migrated first. This argument ultimately returns to the point made earlier that cloud migrations are not purely based on technical considerations, but that they should be made with the impact on the whole business in mind.

8.3.3 Migration Strategies

Once the decision about the migration of a specific application into the cloud is made, there are six common migration strategies, often referred to as "the 6 R's[10]: Re-host, Re-platform, Re-factor, Re-purchase, Retire or Retain (see Figure 8.1, source: Medium website [11])):

- **Re-host:** In this strategy, which is also referred to as "lift and shift", applications are moved without changes from physical in-house servers onto equivalent virtual machines in the cloud. This is obviously a relatively straightforward approach that can be done pretty quickly. One of the disadvantages is that it tends to miss out on many of the advantages that more cloud native deployment approaches can bring, such as reduced manual maintenance effort, higher agility etc.

[10]AWS Migration Whitepaper: `https://docs.aws.amazon.com/whitepapers/latest/aws-migration-whitepaper/welcome.html`

[11]Medium website, `https://shorturl.at/jlBCG`

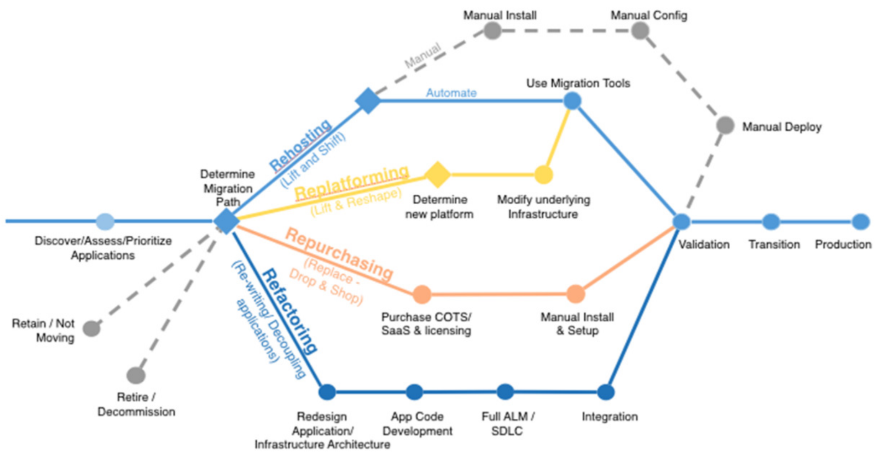

Figure 8.1: Common cloud migration strategies

- **Re-platform:** Also called "lift, thinker and shift", the re-platform approach leaves the overall architecture of the application unchanged. However, in contrast to the re-host approach, some of the underlying platform elements are changed. For example, rather than operating a relational database management system in a customer managed virtual machine, a cloud-provider managed relational database system (such as GCP Cloud SQL) could be used. While such a change likely requires some more development and testing effort than a simple re-hosting, it can lead to additional benefits such as reduced costs (e.g. for software licenses) and reduced ongoing operations effort.

- **Re-factor:** In the re-factor or re-architect approach, the architecture of the application (which is often monolithic to begin with) is changed into a more cloud native architecture, such as a microservice architecture deployed in a Kubernetes cluster. This migration strategy costs the most time and effort. However, once the migration is complete, all the benefits of modern cloud architectures can be realized, including reduced ongoing costs (both for cloud resources and system operations), a highly flexible and easily changeable application, and maximum performance and fault tolerance.

- **Re-purchase:** In this approach, an application that could either be a custom or off-the-shelf software system that was previously operated in-house is replaced by a Software as a Service (SaaS) offering. This only works if there is a SaaS application that matches the needs of the organization well. Even then, it might still be necessary to adjust some business process to the software (rather than the other way around). The big benefit is that the cloud provider takes care of almost all aspects of running the application, which means that in-house IT at most needs to deal with a few aspects such as managing user accounts.

- **Retire:** The remaining two approaches are not strictly migration strategies, because they do not actually result in the application being migrated into the cloud. One possible outcome is that the careful analysis of all existing in-house applications in the course of a cloud migration assessment reveals that an application can actually be retired. This might sound surprising to readers who have no first-hand experience with the often vast IT landscapes of especially larger organizations. However, especially in older organizations, or organizations that are the result of a number of mergers and acquisitions, it is very common to find that some applications are for instance no longer used at all, or only used by users who are not aware that a different, newer application fulfills the same purpose even better. Retiring such legacy applications reduces costs and the complexity of the overall IT in the organization.

- **Retain:** The second migration strategy that does not result in the actual migration of an application into the cloud is to retain the application as it is. There are multiple reasons why this might be the best choice at least for the time being - of course, this decision can be revisited in the future, and the application can be migrated into the cloud then. A common reason to retain an application is that the expected benefits of migrating it into the cloud are less than those of other applications, which consequently, in the face of limited resources, get moved first. Another reason might be that an application is very critical for an organization, and that a move at this point in time (possibly with limited cloud experience) is deemed too risky.

No matter what migration strategy is chosen, moving an application into the cloud consists of multiple stages. Following the Systems Development Life Cycle (SDLC)[12] model, these stages typically include planning, analysis, design, implementation, testing and maintenance stages. They are not usually performed in a single, linear way. Instead, the various stages often overlap, and might be performed multiple times in an iterative fashion,

[12]https://en.wikipedia.org/wiki/Systems_development_life_cycle

depending on the specific project management and software development methods chosen.

Discussing these stages in detail is beyond the scope of this chapter and book. However, we briefly cover two aspects that are often of particular interest in cloud migration projects, namely the migration of application data and the cutover strategy chosen.

Migrating the existing data of an application into the cloud deserves specific attention already in the analysis phase. As mentioned above, the analysis phase needs to pay specific attention to regulatory and other privacy and security requirements. A common scenario is that both the migration of data into the cloud and its storage there needs to be more protected e.g. by encryption than is necessary when data is kept in-house. Applications with large volumes of data can create additional challenges, because even if the organization has a high-bandwidth upload connection to the cloud provider, transferring Terabytes or even Petabytes of data can take impractically long. If this is the case, the data can be transferred by shipping storage media. For example, AWS operates special trucks full of hard disks that can transfer 100 Petabytes at once[13].

Cutover strategies describe how the switch from the in-house to the cloud-based application is performed. The simplest, but also riskiest strategy is a direct cutover, in which at a specific point in time the in-house application is turned off and all traffic is switched over to the new cloud-based application. The main disadvantage of this approach is that it is rather risky. Any problems missed in testing can potentially affect all users of the application at the same time, which might be more than the support can handle. Therefore, some sort of parallel operation between the old in-house system and the new cloud based system is often chosen instead. A common approach is to have a pilot phase in which the cloud based system is initially only used by a small percentage of users, which is increased to all users over a period of days or weeks, depending on how many problems occur. Alternatively, if the architecture of the application permits it, only parts of the application might be moved to the cloud at first, and the remaining parts are migrated step-by step until the whole application runs in the cloud, once again at a speed determined by how long it takes to fix whatever problems are found at each stage.

[13]https://aws.amazon.com/snowmobile/

8.4 A Critical View of Cloud Computing

At this point in the book, readers might have the impression that the authors generally believe cloud computing is an important and useful technology. If so, they would be correct. However, just like any other innovation or technology, cloud computing has its potential dark sides as well. We look at some of them in this chapter.

8.4.1 Common Cloud Computing Challenges

A number of common issues that organizations face when they begin adopting cloud computing has already been discussed in the first chapter. They include:

- Security and privacy concerns - because applications, especially on a public cloud, tend to be more exposed to potential attackers.
- Vendor lock-in - because many cloud computing features are proprietary, it can be difficult to move applications from one cloud provider to another.
- Cost and cost management - while elasticity is a great feature to scale applications based on demand, costs for cloud resources scale with them, which can result in significant and potentially unexpectedly high costs.
- Lack of expertise - because wide-spread adoption of cloud computing is still a relatively recent phenomenon, demand for employees who are trained in and experienced with cloud computing typically outstrips supply.
- Organizational issues - to fully leverage the potential of cloud computing, significant organizational changes are necessary, which often face resistance by employees.

These and similar issues are commonly encountered by organizations, irrespective of factors like the industry the organization operates in or the kinds of applications that are migrated into the cloud. A number of ways to address these issues have already been covered in other parts of the book - e.g. ways to secure cloud applications in the chapter on cloud security, ways to reduce vendor lock-in risks like using multi-cloud deployments in the chapter on multi-cloud systems, and several of the other issues raised in the chapter on cloud migration.

8.4.2 Cloud Computing Limitations

While the issues covered in the previous section apply in most situations, there are specific types of organizations and applications that face additional challenges in a cloud computing setting. In some cases, these challenges can effectively make the use of cloud computing, especially public cloud computing, infeasible.

Organizations that operate in industries with particularly heavy regulatory burdens or security requirements might find adopting cloud computing either undesirable or impossible. For example, government organizations such as police, military or intelligence agencies that are potential targets of highly skilled foreign adversaries are likely to find the security risks of deploying applications especially in a public cloud too high. Similarly, organizations that either handle particularly sensitive information such as patient data in the case of hospitals, or organizations that operate critical infrastructures such as water, energy and telecommunications often face very stringent security requirements that might be hard to fulfill in a cloud setting. These types of regulations often include requirements to only use certified hardware and software products, to have specially trained employees, and to be in full control of where critical data is stored. While some of these requirements can be fulfilled even by public cloud providers, there are often practical limitations.

Industry lobby groups tend to paint government regulations as unnecessary burdens that obstruct businesses unduly, and might not even be effective at reaching their goals. Some populist politicians build on such sentiments when they promise to "cut unnecessary red tape"[14]. While there are certainly instances where this is true, recent incidents like the successful ransomware attack on an oil pipeline operator in the USA[15] that resulted in gasoline shortages in significant parts of the country certainly illustrate that a lack of effective and enforced regulations can lead to organizations neglecting sufficient protection of critical infrastructures e.g. in order to save costs. This in turn can cause widespread harm to other organizations and individuals.

Applications that either transfer very large amounts of data, that require particularly low latencies or have especially stringent requirements for guaranteed response times are also often hard or impossible to implement in the cloud. For example, imagine a manufacturing company that

[14]See e.g. https://en.wikipedia.org/wiki/Red_tape

[15]https://www.cpomagazine.com/cyber-security/colonial-fuel-pipeline-ransomware-attack-that-caused-gas-shortages-in-eastern-u-s-may-be-the-work-of-amateurs/

uses a large number of sensors, including visual sensors such as video cameras, to supervise and control its automated production processes. In such a scenario, it might be inefficient or impossible to upload all the real-time data to the cloud to process it there, because the necessary upload capacity is either too expensive or simply not available. And even if all the data can be uploaded, for time critical applications like automatic discovery of e.g. a robot arm that is about to injure an employee due to an erroneous movement it might take too long to do so and to wait for the return transmission of the results.

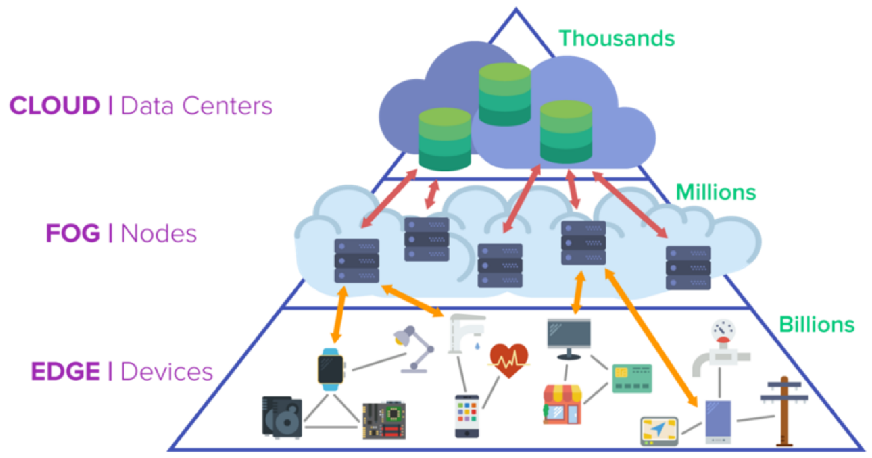

Figure 8.2: Fog, cloud and edge computing.

For these types of applications, local extensions of cloud computing can often be beneficial. Two commonly used terms in this context are Fog and Edge computing (see Figure 8.2, source: omnisci website [16]). Like the term implies, fog computing refers to having cloud-like computing resources (e.g. mini data centers) much closer to the end user. In our example, the manufacturing company might do most of the processing in a small in-house data center that is co-located with its factory, to minimize latencies and to utilize the high bandwidth of local area networks. Edge computing even goes one step further, where (at least some of) the processing happens on the actual devices that create or use the data. In our example, the safety cameras might have sufficient processing power to analyze the live video streams in real time themselves, and might only transfer results (e.g. "danger" or "no danger") to centralized fog or cloud computing resources.

[16] https://www.omnisci.com/technical-glossary/fog-computing

8.5 Ethical Issues in Cloud Computing

This final section looks into situations in which the use of cloud computing might be beneficial for an organization, but might create other, often non-monetary harm to other stakeholders.

8.5.1 Introduction to Ethics in Computing

The concept of ethics can be defined as the "moral principles that govern a person's behaviour or the conducting of an activity" or the "branch of knowledge that deals with moral principles."[17]

A minimum threshold for ethical behaviour is often that laws are followed - or to put this differently, behaviour that breaks laws (e.g. stealing) is usually also unethical. However, not all legal behaviour is also ethical. For example, in many contexts, lying would be considered unethical. However, while there are situations in which lying is actually illegal (e.g. when testifying under oath), there is generally now law prohibiting telling falsehoods.

The fact that just following the law is not sufficient for ethical behaviour is particularly true in connection with technical innovations. The following anecdote can illustrate this. When electricity first became commercially available, some people started to secretly hook up their machines to wires transporting electricity from power plants to paying customers, without paying themselves. When they were caught, attempts to convict them of the crime of theft apparently failed, because the crime of theft is in most jurisdictions defined as something along the lines of the act of taking a physical thing from its rightful owner without permission. However, these "electricity thieves" did not take any physical object from anybody. More recent examples of similar cases where laws were slow to follow technology are areas like copyright and computer security, where early copiers of commercial software or hackers gaining access to computers without authorization might in many jurisdictions not actually have been breaking any laws.

There are some frameworks for deciding what ethical behaviour in computing in general could be. Maybe most notably, the Association for Computing Machinery (ACM) adopted the "ACM Code of Ethics and Professional Conduct"[18] in 1992. It defines a number of general principles that

[17]Definition from Oxford Languages (https://www.google.com/search?q=ethics)
[18]https://www.acm.org/code-of-ethics

computing professionals should follow, such as respecting privacy, honoring confidentiality and knowing and following application laws and regulations.

While such guidelines apply to cloud computing and cloud computing professionals as well, there are several specific areas in cloud computing in which new ethical questions arise. Some of them are briefly discussed in the following sections.

8.5.2 Amplification of Existing Negative Effects of Technology

Many of the applications operated in the cloud could also be operated in non-cloud, on premise data centers. They are often run in the cloud because doing so reduces costs (especially for very elastic applications), increases performance or reliability. Consequently, if such applications cause ethical issues when they are operated in-house, these issues are typically amplified when they are migrated to the cloud, because it becomes economical for more organizations to use them to a greater extent.

An example of this effect is the use of applications using Artificial Intelligence (AI) in the cloud, for example to automatically recognize the content of images and to tag them accordingly. It turns out that such image recognition services are often biased, for example with regard to gender. For example, when presented with images of men and women with the same position (e.g. members of the US senate), men are more likely tagged with labels regarding their role (e.g. business person or official), while women are more likely tagged with superficial aspects of their appearance (e.g. hairstyle or beauty)[19]. While this specific example might not create direct harm to anybody, AI algorithms used e.g. for decisions such as whether to grant a loan to or hire a person based on available data can create very tangible harm for minorities that might be discriminated against because the algorithm relies on biased data.

8.5.3 Privacy Risks

Privacy risks are in some ways just a special example of the previous point, because privacy can of course also be endangered by bad applications operated in-house. But once again, the widespread and inexpensive availability of cloud computing significantly amplifies the issue.

[19]https://www.wired.com/story/ai-sees-man-thinks-official-woman-smile/

The concept of online or data privacy can be best understood by looking at the concept of "Informational Self-Determination" that exists in Germany[20]. This right essentially means that individuals should be able to decide which information about themselves they reveal to others (e.g. to corporations), and for what purposes this information is used. This kind of self-determination does de facto not exist for users of online services. One of many examples is that social network providers and companies involved in placing advertisements on websites can typically track users across different websites. This is highly profitable, because it allows the creation of detailed profiles about individuals, which in turn allows targeted placement of advertisements, which pays a lot more than ads placed on websites randomly. As mentioned above, such privacy risks do not entirely depend on the use of cloud computing. However, the use of cloud computing makes it a lot more feasible (e.g. easier and cheaper) to collect and combine data about individuals from multiple sources, and to run advanced analyses on the data, for example for targeted marketing purposes.

A frequently made argument is that such privacy intrusions are not truly harmful. In fact, they can be beneficial to individuals, because they for example allow for more targeted (and consequently, relevant or interesting) ads, or because it is a reasonable tradeoff for individuals to give up some of their privacy in exchange for online services they do not have to pay for, such as the use of social media platforms. To make this argument in its probably bluntest form: If you are doing nothing illegal, you have nothing to worry about.

Unfortunately, this is not always true. A fairly unusual example showing that even the use of anonymized data can have unexpected consequences is when a fitness tracker app released global maps showing where users of its app were exercising[21]. Even though the released maps contained no data through which individual users could be identified, the mere knowledge that some users were exercising in specific locations created significant security risks. Frequently used jogging routes near an American military base in Afghanistan showed up in the map - since these were almost certainly used by American soldiers, knowledge of these specific routes put them at risk of terrorist attacks.

Another point that could be made is that most people in fact do break the law regularly, mostly by minor offenses, such as driving slightly above

[20]https://medium.com/golden-data/history-of-informational-self-determination-1abe03f98dc1

[21]https://www.theguardian.com/world/2018/jan/28/fitness-tracking-app-gives-away-location-of-secret-us-army-bases

the speed limit, by dropping passengers off in a non-stopping zone, or by not coming to a complete halt at every stop sign. These kinds of very minor offenses are usually not prosecuted, because it would require massive police resources and not be in the best public interest. However, the more data is collected about people online (e.g. by dash cams that upload recordings into the cloud), the easier it becomes to actually punish such offenses. This bears significant potential for abuse by law enforcement authorities, who could still only enforce punishment for such minor offenses very selectively due to limited resources, and might well end up deciding who to prosecute on a discriminatory basis e.g. against racial minorities.

Furthermore, many people have secrets that they want to keep to themselves, for example about their health, religious or political views or sexual orientation. Imagine users of a fitness tracker who want to use features such as automatic tracking of their heart rate, amounts of exercise and sleep in order to keep track of and improve their health. If all the data stays on a local device under the control of the user, rather than being uploaded into the cloud, they might (justifiably) have more trust in the expectation that this data stays private and is not shared with advertisers or other outside parties, either intentionally or accidentally. Finally, it needs to be considered that cloud technology is available globally, which means that countries with authoritarian regimes that suppress any opposition can also use it. This is especially worth considering in debates about whether encryption mechanisms should be legally required to have back doors so that law enforcement can decrypt messages if authorized by a judge. While this sort of mechanism might sound like a good idea, and could in some instances allow the conviction of a criminal who would have otherwise not been caught, it also needs to be considered that such backdoors will ultimately also be available to totalitarian regimes or criminal organizations, putting legitimate users such as supporters of political opposition movements at risk.

8.5.4 Environmental Impacts

A completely different ethical concern is the environmental impact of cloud computing. While this contains other aspects like pollution created by mining to retrieve the raw materials required for all the physical infrastructure in data centers, the energy consumption to operate cloud data centers often gets particular attention, especially due to the increasingly commonly accepted risks of climate change.

According to a study[22], data centers annually consume about 200 terawatt hours of electricity, or approximately 1% of the overall global consumption of electricity. While this might seem like a relatively low percentage, it could be argued that a lot of the cloud applications used are not strictly necessary, and that the high growth rates of cloud computing could mean that the amount of electricity used could also still climb significantly.

An important counter argument is that the environmental impact of cloud computing should reasonably be compared to alternative ways to achieve the same results. For example, while online video streaming services like Netflix clearly consume a significant amount of electricity, which, depending on how the electricity is generated, could result in a significant amount of emissions, the prior approach - people driving to video rental stores to pick up and drop off movies - might well have had even higher emissions. Similarly, it could be argued that large public cloud data centers likely operate more energy efficiently than cloud service users could when they run their own, smaller, and often more under-utilized in-house data centers.

Still, the fact that cloud computing makes computational resources easier and often cheaper to use could well lead to so much more consumption that the overall environmental impact is larger, which does raise ethical questions, such as whether excessive use of computing resources should be curbed e.g. by higher taxes on electricity consumption or emissions.

8.5.5 Market Concentration Risks

Another potential ethical issue is that cloud computing seems to lead to significant market concentration. Due to the effect of economies of scale, large public cloud providers can operate more efficiently than smaller providers, or than organizations can operate their own IT resources in-house. This leads to increasing consolidation of computing in a small number of very large public cloud providers.

Economic pressures can practically force organizations to operate their critical IT systems with public cloud providers. At the same time, especially since most cloud services are not standardized and contain proprietary offerings, moving between different cloud providers or moving systems back in-house becomes increasingly challenging. This gives cloud providers significant power over their users. For example, if in the case of a dispute, a

[22]https://www.datacenterknowledge.com/energy/study-data-centers-responsible-1-percent-all-electricity-consumed-worldwide

cloud provider threatened to shut down an organization's systems on short notice, this would leave many customer organizations little choice but to concede whatever demands the provider has, because an outage of their systems would often endanger the survival of the organization. While the affected organization could of course use the courts to rectify the situation, this would almost certainly take too long.

Another aspect of the same one-sided dependency on a small number or large public cloud providers is that the three largest cloud providers (AWS, Azure and Google Cloud) are all American companies that are subject to interference by the US government. Especially the often aggressive forein policy approach taken by the recent Trump administration created significant concerns in other countries that the US government could force the large cloud providers to stop providing services to companies or government agencies of specific countries as a powerful weapon in trade or other disagreements. In the European Union, this has led to the realization that digital sovereignty, i.e. the ability to be in control over one's own data, is an important goal. Consequently, in recent years, significant resources and effort has to be put into the GAIA-X[23] initiative which aims to create a European infrastructure for cloud services that is independent and safe from potential foreign interference. Other countries such as Switzerland have similar plans (see e.g. [24]).

8.6 Summary

This chapter discussed a number of different issues around operating applications in the cloud, beginning with cloud management, monitoring and migration, and ending with a critical view of cloud computing challenges, limitations and ethical concerns. A unified theme of all these topics is that cloud computing is much more than a purely technical solution. In order to use cloud computing efficiently, effectively and in an ethical way, appropriate organizational measures need to be taken as well. This is a good way to end this book, which, while primarily intended for an audience with some background and interest in computing, should also be useful for readers with an interest in a broader perspective of the topic.

[23]https://www.data-infrastructure.eu/GAIAX/Navigation/EN/Home/home.html
[24]https://swissdatasovereignty.ch/