1

Introduction to Multi-Cloud

Multi-cloud is a hot topic with companies. Most companies are already multi-cloud, sometimes even without realizing it. They have **Software as a Service (SaaS)** such as Office 365 from Microsoft and Salesforce, for instance, next to applications that they host in a public cloud such as **Amazon Web Services (AWS)** or **Google Cloud Platform (GCP)**. It's all part of the digital transformation that companies are going through, that is, creating business agility by adopting cloud services where companies develop a best-of-breed strategy: picking the right cloud service for specific business functions. The answer might be multi-cloud, rather than going for a single cloud provider.

The main goal of this chapter is to develop a foundational understanding of what multi-cloud is and why companies have a multi-cloud strategy. We will focus on the main public cloud platforms of Microsoft Azure, **AWS**, **and GCP**, next to the different on-premises variants of these platforms, such as Azure Stack, AWS Outposts, Google Anthos, and some emerging players.

The most important thing before starting the transformation to multi-cloud is gathering requirements, making sure a company is doing the right thing and making the right choices. Concepts such as **The Open Group Architecture Framework (TOGAF)** and **Quality Function Deployment (QFD)** will be discussed as tools to capture the **voice of the customer (VOC)**. Lastly, you will learn that any transformation starts with people. The final section discusses the changes to the organization itself needed to execute the digital transformation.

In this chapter, we're going to cover the following main topics:

- Understanding multi-cloud concepts
- Multi-cloud—more than just public and private
- Setting out a real strategy for multi-cloud

- Introducing the main players in the field
- Evaluating cloud service models
- Gathering requirements for multi-cloud
- Understanding the business challenges of multi-cloud

Understanding multi-cloud concepts

This book aims to take you on a journey along the different major cloud platforms and will try to answer one crucial question: *if my organization deploys IT systems on various cloud platforms, how do I keep control?* We want to avoid cases where costs in multi-cloud environments grow over our heads, where we don't have a clear overview of who's managing the systems, and, most importantly, where system sprawl introduces severe security risks. But before we start our deep-dive, we need to agree on a common understanding of *multi-cloud* and multi-cloud concepts.

There are multiple definitions of multi-cloud, but we're using the one stated at `https://www.techopedia.com/definition/33511/multi-cloud-strategy`:

> *Multi-cloud refers to the use of two or more cloud computing systems at the same time. The deployment might use public clouds, private clouds, or some combination of the two. Multi-cloud deployments aim to offer redundancy in case of hardware/software failures and avoid vendor lock-in.*

Let's focus on some topics in that definition. First of all, we need to realize where most organizations come from: traditional datacenters with physical and virtual systems, hosting a variety of functions and business applications. If you want to call this *legacy*, that's OK. But do realize that the cutting edge of today is the legacy of tomorrow. Hence, in this book, we will refer to "traditional" IT when we're discussing the traditional systems, typically hosted in physical, privately owned datacenters. And with that, we've already introduced the first problem in the definition that we just gave for multi-cloud.

A lot of enterprises call their virtualized environments private clouds, whether these are hosted in external datacenters or in self-owned, on-premises datacenters. What they usually mean is that these environments host several business units that get billed for consumption on a centrally managed platform. You can have long debates on whether this is really using the cloud, but the fact is that there is a broad description that sort of fits the concept of private clouds.

Of course, when talking about the cloud, most of us will think of the major public cloud offerings that we have today: AWS, Microsoft Azure, and GCP. These are public clouds: providers that offer IT services on demand from centralized platforms using the public internet. They are centralized platforms that provide IT services such as compute, storage, and networking but distributed across datacenters around the globe. The cloud provider is responsible for managing these datacenters and, with that, the cloud. Companies "rent" the services, without the need to invest in datacenters themselves.

By another definition, multi-cloud is a best-of-breed solution from these different platforms, creating added value for the business in combination with this solution and/or service. So, using the *cloud* can mean either a combination of solutions and services in the public cloud or combined with private cloud solutions.

But the simple feature of combining solutions and services from different cloud providers and/or private clouds does not make up the multi-cloud concept alone. There's more to it.

Maybe the best way to explain this is by using the analogy of the smartphone. Let's assume you are buying a new phone. You take it out of the box and switch it on. Now, what can you do with that phone? First of all, if there's no subscription with a telecom provider attached to the phone, you will discover that the functionality of the device is probably very limited. There will be no connection from the phone to the outside world, at least not on a mobile network. An option would be to connect it through a Wi-Fi device, if Wi-Fi is available. In short, one of the first actions, in order to actually use the phone, would be making sure that it has connectivity.

Now you have a brand-new smartphone set to its factory defaults and you have it connected to the outside world. Ready to go? Probably not. You probably want to have all sorts of services delivered to your phone, usually through the use of apps, delivered through online catalogs such as an app store. The apps themselves come from different providers and companies, including banks and retailers, and might even be coded in different languages. Yet, they will work on different phones with different versions of mobile operating systems such as iOS or Android.

You will also very likely want to configure these apps according to your personal needs and wishes. Lastly, you need to be able to access the data on your phone. All in all, the phone has turned into a landing platform for all sorts of personalized services and data.

The best part is that in principle, you, the user of the phone, don't have to worry about updates. Every now and then the operating system will automatically be updated and most of the installed apps will still work perfectly. It might take a day or two for some apps to adapt to the new settings, but in the end, they will work. And the data that is stored on the phone or accessed via some cloud directory will also still be available. The whole ecosystem around that smartphone is designed in such a way that from the end user's perspective, the technology is completely transparent:

Figure 1.1: Analogy of the smartphone—a true multi-cloud concept

Well, this mirrors the concept of the cloud, where the smartphone in our analogy is the actual integrated landing zone, where literally everything comes together, providing a seamless user experience.

How is this an analogy for multi-cloud? The first time we enter a portal for any public cloud, we will notice that there's not much to see. We have a platform—the cloud itself—and we probably also have connectivity through the internet, so we can reach the portal. But we don't want everyone to be able to see our applications and data on this platform, so we need to configure it for our specific usage. After we've done that, we can load our applications and the data on to the platform. Only authorized people can access those applications and that data. However, just like the user of a smartphone, a company might choose to have applications and data on other platforms. They will be able to connect to applications on a different platform.

The company might even decide to migrate applications to a different platform. Think of the possibility of having Facebook on both an iPhone and an Android phone; with just one Facebook account, the user will see the same data, even when the platforms—the phones—use different operating systems.

Multi-cloud—more than just public and private

There's a difference between hybrid IT and multi-cloud, and there are different opinions on the definitions. One is that hybrid platforms are homogeneous and multi-cloud platforms are heterogeneous. *Homogeneous* here means that the cloud solutions belong to one stack, for instance, the Azure public cloud with Azure Stack on-premises. *Heterogeneous*, then, would mean combining Azure and AWS, for instance.

Key definitions are:

- **Hybrid**: Combines on-premises and cloud.
- **Multi-cloud**: Two or more cloud providers.
- **Private**: Resources dedicated to one company or user.
- **Public**: Resources are shared (note, this doesn't mean anyone has access to your data. In the public cloud, we will have separate tenants, but these tenants will share resources, for instance, in networking).

For now, we will keep it very simple: a hybrid environment combines an on-premises stack—a private cloud—with a public cloud. It is a very common deployment model within enterprises and most consultancy firms have concluded that these hybrid deployments will be the most implemented future model of the cloud.

Two obvious reasons for hybrid—a mixture between the public and private clouds—are security and latency, besides the fact that a lot of companies already had on-premises environments before the cloud entered the market.

To start with security: this is all about sensitive data and privacy, especially concerning data that may not be hosted outside a country, or outside certain regional borders, such as the **European Union** (**EU**). Data may not be accessible in whatever way to—as an example—US-based companies, which in itself is already quite a challenge in the cloud domain. Regulations, laws, guidelines, and compliance rules often prevent companies from moving their data off-premises, even though public clouds offer frameworks and technologies to protect data at the very highest level. We will discuss this later on in *Part 4* of this book in *Chapters 13* to *18*, where we talk about security, since security and data privacy are of the utmost importance in the cloud.

Latency is the second reason to keep systems on-premises. One example that probably everyone can relate to is that of print servers. Print servers in the public cloud might not be a good idea. The problem with print servers is the spooling process. The spooling software accepts the print jobs and controls the printer to which the print assignment has to be sent. It then schedules the order in which print jobs are actually sent to that printer. Although print spoolers have improved massively in recent years, it still takes some time to execute the process. Print servers in the public cloud might cause delays in that process. Fair enough: it can be done, and it will work if configured in the right way, in a cloud region close to the sending PC and receiving printer device, plus accessed through a proper connection.

You get the idea, in any case: there are functions and applications that are highly sensitive to latency. One more example: retail companies have warehouses where they store their goods. When items are purchased, the process of order picking starts. Items are labeled in a supply system so that the company can track how many of a specific item are still in stock, where the items originate from, and where they have to be sent. For this functionality, items have a barcode or QR code that can be scanned with RFID or the like. These systems have to be close to the production floor in the warehouse or—if you do host them in the cloud—accessible through really high-speed, dedicated connections on fast, responsive systems.

These are pretty simple and easy-to-understand examples, but the issue really comes to life if you start thinking about the medical systems used in operating theatres, or the systems controlling power plants. It is not that useful to have an all-public-cloud, cloud-first, or cloud-only strategy for quite a number of companies and institutions. That goes for hospitals, utility companies, and also for companies in less critical environments.

Yet, all of these companies discovered that the development of applications was way more agile in the public cloud. Usually, that's where cloud adoption starts: with developers creating environments and apps in public clouds. It's where hybrid IT is born: the use of private systems in private datacenters for critical production systems that host applications with sensitive data that need to be on-premises for latency reasons, while the public cloud is used to enable the fast, agile development of new applications. That's where new cloud service models come into the picture. These models are explored in the next section.

The terms multi-cloud and hybrid get mixed up a lot and the truth is that a solution can be a mix. You can have, as an example, dedicated private hosts in Azure and AWS, hence running private servers in a public cloud. Or, run cloud services on a private host that sits in a private datacenter, for instance, with Azure Stack or AWS Outposts. That can lead to confusion. Still, when we discuss hybrid in this book, we refer to an on-premises environment combined with a public cloud. Multi-cloud is when we have two or more cloud providers.

CaaS

A growing number of enterprises are adopting container technology to host, run, and scale their applications. To run containers, developers must set up a runtime environment for these containers. Typically, this is done with Kubernetes, which has developed as the industry standard to host, orchestrate, and run containers. Setting up Kubernetes clusters can be complex and time-consuming. **Container as a Service (CaaS)** is the solution. CaaS provides an easy way to set up container clusters.

XaaS

Anything as a Service (XaaS) is a term used to express the idea that users can have everything as a service. The concept is widely spread with, for instance, **Hardware as a Service (HaaS)**, **Desktop as a Service (DaaS)**, or **Database as a Service (DBaaS)**. This is not limited to IT, though. The general idea is that companies will offer services and products in an *as a service* model, using the cloud as the digital enabler. Examples are food delivery to homes, ordering taxis, or consulting a doctor using apps.

Although we will touch upon SaaS and containers, we will focus mainly on IaaS and PaaS as starting points to adopt multi-cloud. With that in mind, we can start by setting out our multi-cloud strategy.

Setting out a real strategy for multi-cloud

A cloud strategy emerges from the business and the business goals. Business goals, for example, could include the following:

- Creating more brand awareness
- Releasing products to the market faster
- Improving profit margins

Business strategies often start with increasing revenue as a business goal. In all honesty: that should indeed be a goal; otherwise, you'll be out of business before you know it. The strategy should focus on *how* to generate and increase revenue. We will explore more on this in *Chapter 2*, *Business Acceleration Using a Multi-Cloud Strategy*.

How do you get from business goals to defining an IT strategy? That is where enterprise architecture comes into play. The most used framework for enterprise architecture is **TOGAF**, although there are many more frameworks that can be used for this. Companies will not only work with TOGAF but also adopt practices from IT4IT and ITIL to manage their IT, as examples.

The core of TOGAF is the **ADM** cycle, short for **Architecture Development Method**. Also, in architecting multi-cloud environments, ADM is applicable. The ground principle of ADM is **B-D-A-T**: the cycle of **business, data, applications, and technology**. This perfectly matches the principle of multi-cloud, where the technology should be transparent. Businesses have to look at their needs, define what data is related to those needs, and how this data is processed in applications. This is translated into technological requirements and finally drives the choice of technology, integrated into the architecture vision as follows:

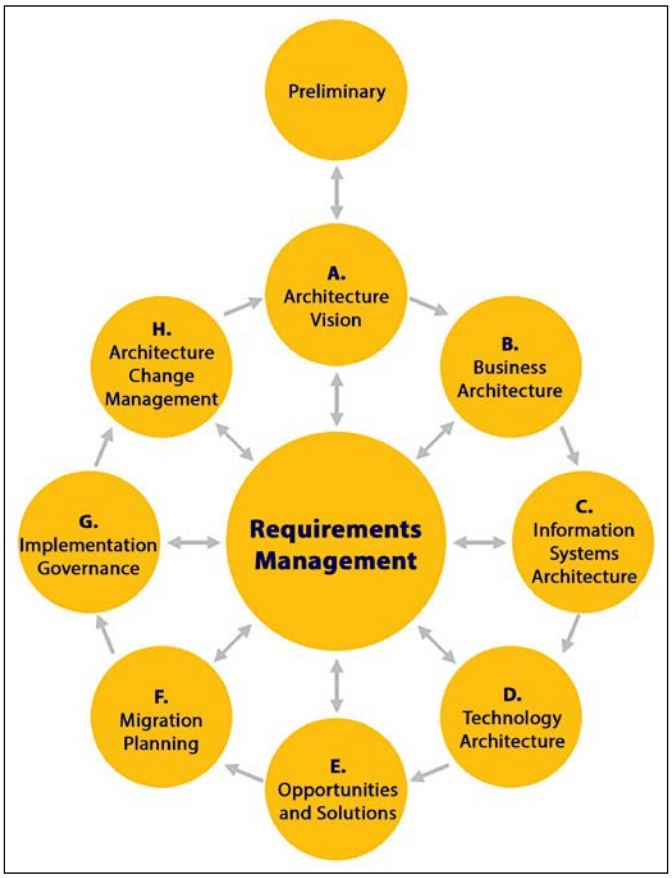

Figure 1.3: The ADM cycle in the TOGAF

This book is not about TOGAF, but it does make sense to have knowledge of enterprise architecture and, for that matter, TOGAF is the leading framework. TOGAF is published and maintained by The Open Group. More information can be found at `https://www.opengroup.org/togaf`.

The good news is that multi-cloud offers organizations flexibility and freedom of choice. That also brings a risk: lack of focus, along with the complexity of managing multi-cloud. Therefore, we need a strategy. Most companies adopt cloud and multi-cloud strategies since they are going through a process of transformation from a more-or-less traditional environment into a digital future. Is that relevant for all businesses? The answer is yes. In fact, more and more businesses are coming to the conclusion that IT is one of their core activities.

Times have changed over the last few years in that respect. At the end of the nineties and even at the beginning of the new millennium, a lot of companies outsourced their IT since it was not considered to be a core activity. That has changed dramatically over the last 10 years or so. Every company is a software company—a message that was rightfully quoted by Microsoft CEO Satya Nadella, following an earlier statement by the father of software quality, Watts S. Humphrey, who already claimed at the beginning of the millennium that every business is a software business.

Both Humphrey and Nadella are right. Take banks as an example: they have transformed to become more and more like IT companies. They deal with a lot of data streams, execute data analytics, and develop apps for their clients. A single provider might not be able to deliver all of the required services, hence these companies look for multi-cloud, best-of-breed solutions to fulfill these requirements.

These best-of-breed solutions might contain traditional workloads with a classic server-application topology, but will more and more shift to the use of PaaS, SaaS, container, and serverless solutions in an architecture that is more focused on microservices and cloud-native options. This has to be considered when defining a multi-cloud strategy: a good strategy would not be "cloud first" but "cloud-fit," meaning that a good strategy would not be about multi-cloud or single cloud, but having the right cloud.

Gathering requirements for multi-cloud

One of the first things that companies need to do is gather requirements before they start designing and building environments on cloud platforms. The most important question is probably: what will be the added value of using cloud technology to the business? Enterprises don't move to the cloud because they can, but because cloud technology can provide them with benefits. Think about not only agility, speed, and flexibility in the development of new services and products but also financial aspects: paying only for resources that they actually use or because of automation being able to cut costs in operations.

2

Collecting Business Requirements

This chapter discusses how enterprises can accelerate business results by implementing a multi-cloud strategy. Typically, this is a task for enterprise or business architects, but in digital transformation, enterprise architecture and cloud architecture are tightly connected. We have to collect business requirements as a first step before we can think of the actual cloud strategy.

Every cloud platform/technology has its own benefits and by analyzing business strategies and defining what cloud technology fits best, enterprises can really take advantage of multi-cloud. A strategy should not be "cloud-first" but "cloud-fit." But before we get into the technical strategy and the actual cloud planning, we must explore the business or enterprise strategy and the financial aspects that drive this strategy.

In this chapter, we're going to cover the following main topics:

- Analyzing the enterprise strategy for the cloud
- Defining the cloud strategy from the enterprise architecture
- Fitting cloud technology to business requirements
- Applying the value streams of IT4IT
- Keeping track of cloud developments—focusing on the business strategy
- Creating a comprehensive business roadmap
- Mapping the business roadmap to a cloud-fit strategy

Analyzing the enterprise strategy for the cloud

Before we get into a cloud strategy, we need to understand what an enterprise strategy is and how businesses define such a strategy. As we learned in the previous chapter, every business should have the goal of generating revenue and earning money. That's not really a strategy. The strategy is defined by how it generates money with the products the business makes or the services that it delivers.

A good strategy comprises a well-thought-out balance between timing, access to and use of data, and something that has to do with braveness—daring to make decisions at a certain point in time. That decision has to be based on—you guessed it—proper timing, planning, and the right interpretation of data that you have access to. If a business does this well, it will be able to accelerate growth and, indeed, increase revenue. The overall strategy should be translated into use cases. Use cases can be:

- Delivering products in new business models, such as SaaS
- Achieving more resilience in business, for instance, by implementing disaster recovery using the cloud
- Faster time to market in product development through the quick deployment of development environments
- Analysis of big data using data lakes in the cloud

These use cases must be reflected by the strategy of the enterprise. They will drive the decisions in designing and implementing cloud solutions and even in the choice of cloud platforms.

The success of a business is obviously not only measured in terms of revenue. There are a lot of parameters that define success as a whole and for that matter, these are not limited to just financial indicators. Nowadays, companies rightfully also have social indicators to report on. Think of sustainability and social return. However, a company that does not earn money, one way or the other, will likely not last long.

What are the drivers for business strategy? Typically, these are categorized into four areas:

- Financial objectives
- Customer objectives
- Product objectives
- Internal objectives

In the first chapter, the customer requirements were discussed alongside the methodologies to capture these, the use of **Quality Function Deployment (QFD)**, **the House of Quality (HOQ)**, **and the Voice of the Customer (VOC)**. By understanding the customer needs, the enterprise is able to design, develop, and deploy products that customers are willing to buy—if the price is right. So, there's another challenge for the enterprise: it needs to deliver products at a price that is acceptable to customers. The price needs to cover the costs and preferably with enough margin to make some profit. However, there's one more aspect that is crucial to enterprise strategy: timing.

Time is one of the most important factors to consider when planning for business acceleration. Having said that, it's also one of the most difficult things to grasp. No one plans for a virus outbreak and yet it happened in 2020, leading to a worldwide pandemic. It was a reason for businesses not to push for growth at that time. The strategy for a lot of companies probably changed from pushing for growth to staying in business by trying to drive costs down. It proved that modern businesses have to be extremely agile.

Business agility has become one of the most important strategic drivers to go to the cloud, but what is business agility? It's the capability of a business to respond quickly to events in rapidly changing markets. This comes with a different enterprise architecture. The enterprise has to become adaptive in every layer. That means that the organization has to change: smaller, agile working teams, interacting closely with customers and focusing on smaller tasks that can be executed in sprints of a couple of weeks.

But it also means that systems must become agile. Changing customer demands must be met quickly, resulting in new features in systems. A lot of enterprises still have a massive technical debt, with big, monolithic systems that are hard to update and upgrade without the need to change the entire system. Modern architecture is about microservices: applications are compiled as a collection of services that are loosely coupled. Teams will work on specific services, independently from other teams. The services will interact with each other through protocols such as HTTP, TCP, and AMQP.

Shifting to a subscription-based economy

In the past decade, the markets for most businesses have changed dramatically. Markets have shifted from privately owned to platform economies and eventually a subscription-based economy. Products and services are used as a subscription, "as a service." Consumers can have services on any device, at any time, at any place they desire. They simply pay a fee to get access to a service; they don't want to own the product or service.

The challenge for businesses is that services are now consumed in a completely different manner. It's not a one-off sale anymore, and customers can actually do something with the subscription: it can be paused, changed, suspended, restated, or stopped completely. Subscriptions are very fluid. Hence, enterprises must have architectures that are capable of addressing this flexibility, and systems must be agile and scalable.

Another aspect crucial to business strategy and agility is access to and the use of data. It looks like the use of data in business is something completely new, but of course, nothing could be further from the truth. Every business in every era can only exist through the use of data. We might not always consider something to be data since it isn't always easy to identify, especially when it's not stored in a central place.

Data is the key. It's not only about raw data that a business (can) have access to, but also about analyzing that data. Where are my clients? What are their demands? Under what circumstances are these demands valid and what makes these demands change? How can I respond to these changes? How much time would I have to fulfill the initial demands and apply these changes if required? This all comes from data. Nowadays, we have a lot of data sources. The big trick is how we can make these sources available to a business—in a secure way, with respect to confidentiality, privacy, and other compliance regulations.

An example will make this clearer. The changes in the global healthcare market make an excellent example of business challenges, due to changing market circumstances such as a lack of skilled staff and a globally aging population, requiring more cure and care. The sustainability of the global health system is under high pressure. Hence, there's a general understanding that there should be more focus on prevention, rather than on treatments. Treatment costs society a lot more than preventing people from getting sick in the first place. Governments and companies are therefore now trying to make sure that people start improving their lifestyles. Data is absolutely essential to start this improvement and then to develop services that will help people develop and maintain a better lifestyle.

Companies are investing in collecting health data. And no surprise, it's big tech that's heading the game, since the place to collect data is the cloud. It's collected from medical devices and also from devices such as smartwatches and equipment in gyms connected to the internet. This data is analyzed so that trends can be made visible. Based on that data, individual health plans can be organized, but the data can also be utilized for commercial goals: selling running shoes, diet products, or health services, preferably on a subscription basis so that services can easily be adapted when customer demands change.

Challenges in this healthcare use case are primarily confidentiality and protection of data, and compliance with international privacy regulations. These are bigger challenges than the technology itself. The technology to collect data from various sources and make it available through data mining and analytics is generally available.

In the following sections, we will address business challenges such as business agility, security, data protection, and time to market and understand why cloud technology can help enterprises in dealing with these challenges.

Considering cloud adoption from enterprise architecture

In the previous section, the changes in the enterprise markets were discussed. The modern economy is changing from ownership to subscription-based models, leading to the need for flexibility, adaptability, agility, and scalability in all layers of the enterprise, including the organization and the systems. Subscriptions come with new payment models such as **pay-as-you-go** (**PAYG**) and freemium concepts, where a basic service is delivered at no charge, but customers can enhance service with paid options. Services must be interoperable, but also capable of interacting with other systems such as payment services.

The overarching enterprise architecture as a result of this digital transformation will inevitably change. The architecture will enable faster development, targeting microservices developed by smaller teams and using cloud-native technology. In the next sections, you will learn what an architect should take into account in defining this new digital, cloud-native strategy. This is typically the work of an enterprise architect but, as we already mentioned in the introduction to this chapter, with digital transformation, enterprise and strategic cloud architecture are getting closely related to each other. The enterprise architect must understand the cloud, and the cloud architect should have knowledge about enterprise architecture since cloud adoption is not purely a technological subject.

Long-term planning

When a business is clear on its position and its core competencies, it needs to set out a plan. Where does the company want to be in 5 years from now? This is the most difficult part. Based on data and data analytics, it has to determine how the market will develop and how the company can anticipate change. Again, data is absolutely key, but so is the swiftness with which companies can change course since market demands do change extremely rapidly.

Introducing the scaffold for multi-cloud environments

How does a business start in the cloud? You would be surprised, but a lot of companies still just start without having a plan. How difficult can it be, after all? You get a subscription and begin deploying resources. That probably works fine with really small environments, but you will soon discover that it grows over your head. Think about it—would you start building a data center just by acquiring a building and obtaining an Ethernet cable and a rack of servers? Of course not. So why would you just start building without a plan in the public cloud? You would be heading for disaster, and that's no joke. As we saw in *Chapter 1*, *Introduction to Multi-Cloud*, a business will need a clear overview of costs, a demarcation of who does what, when, and why in the cloud, and, most importantly, it all needs to be secure by protecting data and assets, just as a business would do in a traditional data center.

If there's one takeaway from this book, it's this: you are building a data center. You are building it using public clouds, but it's a data center. Treat it as a data center.

Luckily, all major cloud providers feel exactly the same way and have issued cloud adoption frameworks. Succinctly put, these frameworks help a business in creating the plan and, first and foremost, help to stay in control of cloud deployments. These frameworks do differ on certain points, but they also share a lot of common ground.

Now, the title of this section contains the word *scaffold*. The exact meaning of **scaffold** is a structure that supports the construction and maintenance of buildings; the term was adopted by Microsoft to support, build, and manage environments that are deployed in Azure. It's quite an appropriate term, although in the cloud, it would not be a temporary structure. It's the structure that is used as the foundation to build and manage the environments in a cloud landing zone.

Scaffolding comprises a set of pillars, which will be covered in the following sections.

Working with Well-Architected Frameworks

Well-Architected Frameworks were invented to help customers build environments in public clouds by providing them with best practices. This book assumes that you are working in multi-cloud and facing the challenge that every cloud provider has its own tools, workflows, set of commands, service naming, and mapping. The good news is that all major cloud providers provide Well-Architected Frameworks. Even better news: these frameworks all do sort of the same thing.

Today, automation is often executed per component or, in the best cases, per platform. By doing that, we're not reaching the final goal of automation, which is to reduce manual tasks that have to be executed over and over again. We're not reducing the human labor and, for that matter, we're not reducing the risk of human error. On the contrary, we are introducing more work and the risk of failure by having automation divided into different toolsets on top of different platforms, all with separate workflows, schedules, and scripts. Hyperautomation deals with that. It automates all business processes and integrates these in a single automated life cycle, managed from one automation platform.

In summary, cloud adoption frameworks from Azure, AWS, and GCP all support the same principles. That's because they share the common IT service management language. That helps us to align processes across platforms. The one challenge that we have is the single dashboard to control the various platforms and have one source of automation across the cloud platforms—hyperautomation. With the speed of innovation in the cloud, that is becoming increasingly complex, but we will see more tools and automation engines coming to market over the next years, including the rise and adoption of AIOps.

Understanding identities and roles in the cloud

Everything in the cloud has an identity. There are two things that we need to do with identities: authenticate and authorize. For authentication, we need an identity store. Most enterprises will use **Active Directory (AD)** for that, where AD becomes the central place to store the identities of persons and computers. We won't be drilling down into the technology, but there are a few things you should understand when working with AD. First of all, an AD works with domains. You can deploy resources—VMs or other virtual devices—in a cloud platform, but if that cloud platform is not part of your business domain, it won't be very useful. So, one of the key things is to get resources in your cloud platform domain-joined. For that, you will have to deploy domain services with domain controllers in your cloud platform or allow cloud resources access to the existing domain services. By doing that, we are extending the business to the cloud platform.

That sounds easier than it is in practice. Azure, AWS, and GCP are public clouds. Microsoft, Amazon, and Google are basically offering big chunks of their platforms to third parties: businesses that host workloads on a specific chunk. But they will still be on a platform that is owned and controlled by the respective cloud providers. The primary domain of the platform will be `onmicrosoft.com` or `aws.amazon.com`: this makes sense if you think of all the (public) services they offer on their platforms.

If we want our own domain on these platforms, we will need to ring-fence a specific chunk by attaching a registered domain name to the platform. Let's, for example, have a company with the name `myfavdogbiscuit.com`. On Azure, we can specify a domain with `myfavdogbiscuit.onmicrosoft.com`. Now we have our own domain on the Azure platform. The same applies obviously to AWS and GCP. Resources deployed in the cloud domains can now be domain-joined, if the domain on the cloud platform is connected to the business domain. That connection is provided by domain controllers. The following diagram shows the high-level concept of AD Federation:

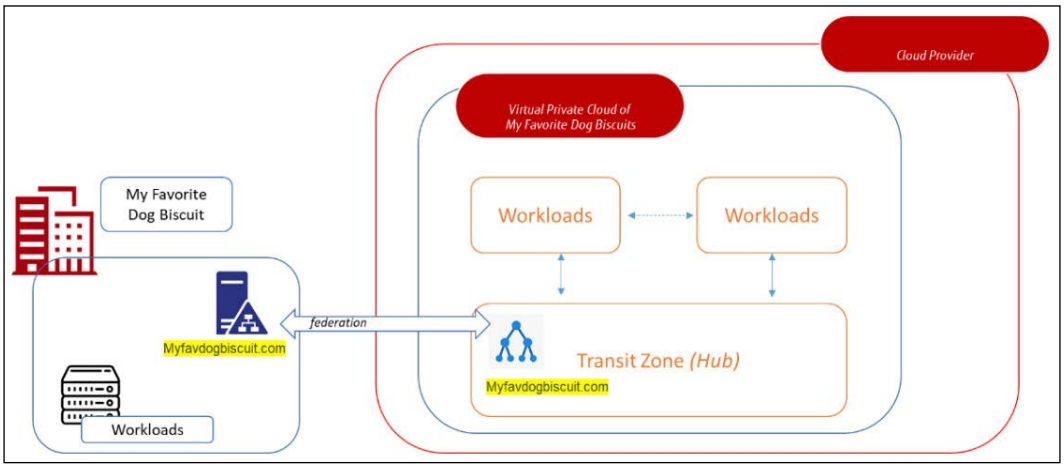

Figure 4.4: Active Directory Federation

In AD, we have all our resources and persons that are allowed inside our domain. Authentication is done through acknowledgment: an identity is recognized in the domain or rejected. This AD uses Kerberos to verify an identity. It's important to know that all cloud providers support AD, the underlying **Lightweight Directory Access Protocol (LDAP)** standard, and Kerberos.

If a resource or person can't be identified in the directory, it simply won't get access to that domain, unless we explicitly grant them access. That can be the case when a person doesn't have an account in our domain, but needs to have access to resources that are on our platform. We can grant this person access using a business-to-business connection. In Azure, that is conveniently called B2B where Azure AD is used for external identities. In AWS it's called Cognito, and in GCP, Cloud Identity.

We have identified a person or a resource using the directory, but now we have to make sure that this resource can only do what we want or allow it to do and nothing more. This is what we call authorization: we specify what a resource is allowed to do, when certain criteria are met. First, we really want to make sure that the resource is whatever it claims it is.

For people logging in, it is advised to use multi-factor authentication. For compute resources, we will have to work with another mechanism and typically that will be based on keys: a complex, unique hash that identifies the resource.

One more time: we have defined identities in our environment, either human personnel or system identities. How can we define what an identity is allowed in our environment? For that, we need **Role-Based Access Control** (**RBAC**). RBAC in Azure, IAM in AWS, and Cloud Identity in GCP let you manage access to (parts of) the environment and what identities can do in that environment. We can also group identities to which a specific RBAC policy applies.

We have already concluded that all cloud platforms support AD and the underlying protocols. So, we can federate our AD with domains in these different clouds. Within Azure, the obvious route to do so would be through connecting AD to Azure AD. Although it might seem like these are similar solutions, they are totally different things. AD is a real directory, whereas Azure AD is a native identity provider within Azure, able to authenticate cloud-native identities. It does not have the authentication mechanisms that AD has, such as Kerberos. Azure AD will authenticate using AD. And with that, it's quite similar to the way the other platforms federate with AD. In both AWS and GCP, you will need identities that can be federated against AD. In other words, your AD will always remain the single source of truth for identity management, the one and only identity store.

Creating the service design and governance model

The final thing to do is to combine all the previous sections into a service design and governance model for multi-cloud environments. So, what should the contents be of a service design? Just look at everything we have discussed so far. We need a design that covers all the topics: requirements, identities and access management, governance, costs, and security. Let's discuss these in detail.

Requirements

This includes the service target that will comprise a number of components. Assuming that we are deploying environments in the public cloud, we should include the public cloud platform as such as a service target. The SLA for Microsoft Online Services describes the SLAs and KPIs committed to by Microsoft for the services delivered on Azure. These are published on `https://azure.microsoft.com/en-us/support/legal/sla/`. For AWS, the SLA documentation can be found at `https://aws.amazon.com/legal/service-level-agreements/`. Google published the SLAs for all cloud services on GCP at `https://cloud.google.com/terms/sla/`. These SLAs will cover the services that are provided by the respective cloud platforms; they do not cater to services that a business builds on top of these cloud-native services.

5

Managing the Enterprise Cloud Architecture

In the previous chapters, we learned about different cloud technology strategies and started drafting a service model, including governance principles. Where do we go from here? From this point onward, you will be—as a business—managing your IT environments in multi-cloud. Successfully managing this new estate means that you will have to be very strict in maintaining the enterprise architecture. Hence, this chapter is all about maintaining and securing the multi-cloud architecture.

This chapter will introduce the methodology to create an enterprise architecture for multi-cloud using **The Open Group Architecture Framework (TOGAF)** and other methodologies, including **Continuous Architecture** and the recently published **Open Agile Architecture**. We will study how to define architecture principles for various domains such as security, data, and applications using quality attributes. We will also learn how we can plan and create the architecture in different stages. Lastly, we will discuss the need to validate the architecture and how we can arrange it.

In this chapter, we will cover the following topics:

- Defining architecture principles for multi-cloud
- Using quality attributes in architecture
- Creating the architecture artifacts
- Change management and validation as the cornerstone
- Validating the architecture

Defining architecture principles for multi-cloud

We'll start this chapter from the perspective of enterprise architecture using the **Architecture Development Method** (**ADM**) cycle in TOGAF as a guiding and broadly accepted framework for enterprise architecture. In *Chapter 2*, *Business Acceleration Using a Multi-Cloud Strategy*, we learned that the production cycle for architecture starts with the business, yet there are two steps before we actually get to defining the business architecture: we have a preliminary phase where we set out the framework and the architecture principles. These feed into the very first step in the actual cycle, known as architecture vision, as shown in the following diagram:

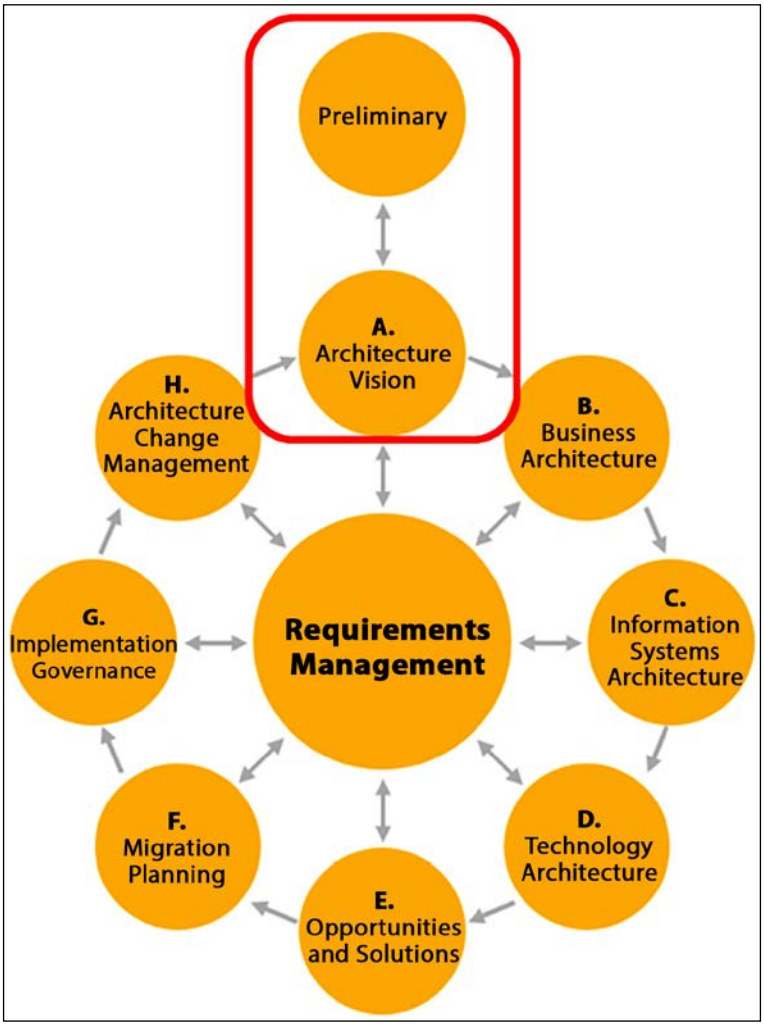

Figure 5.1: The preliminary phase and architecture vision in TOGAF's ADM cycle

The key to any preliminary phase is the architecture principles; that is, your guidelines for fulfilling the architecture. There can be many principles, so the first thing that we have to do is create principle groups that align with our business. The most important thing to remember is that principles should enable your business to achieve its goals. Just to be very clear on this aspect: going to the cloud is not a business goal, just like cloud-first is not a business strategy. These are technology statements at best, nothing more. But principles should do more: they have to support the architectural decisions being made and for that reason, they need to be durable and consistent.

When businesses decide that the cloud might be a good platform to host business functions and applications, the most used principles are flexibility, agility, and cost-efficiency. The latter is already highly ambiguous: what does cost-efficient mean? Typically, it means that the business expects that moving workloads to cloud platforms is cheaper than keeping them on-premises. This could be the case, but the opposite can also be true if incorrect decisions in the cloud have been made by following bad architectural decisions based on ambiguous principles. In short, every principle should be challenged:

- Does the principle support the business goals?
- Is the principle clear so that it can't be subject to multiple interpretations?
- Is the principle leading toward a clearly defined solution?

Some suggested groups for defining principles are as follows:

- Business
- Security and compliance
- Data principles
- Application principles
- Infrastructure and technology principles
- Usability
- Processes

Let's talk about each category in detail, but before we do so, we must learn by what values we are defining the architecture. These values are captured in quality attributes, to be discussed in the next section.

Using quality attributes in architecture

Quality attributes is a term that sprouts from a different framework, called **Continuous Architecture**. Cloud architectures tend to be fluid, meaning that they are very dynamic since they have to respond to changing customer demands fast and continuously. Cloud services are used to enable the business agility that we discussed in the previous chapter. Cloud services, therefore, need to be scalable, but still reliable. Lastly, enterprises do not want to be locked in on any platform, so environments must be able to communicate with other systems or even move to other platforms.

Architectures of systems might change constantly, due to fast developments and continuous new releases enabled by DevOps. Architects have a tedious task to accomplish this. Continuous architecture might be a good guide.

This framework defines quality attributes to which architecture must comply:

- **Operability**: This part of the architecture covers automation in the first place, but also monitoring and logging. In essence, operability is everything that is needed to keep systems operational in a secure state. This requires monitoring:
    - A key decision in monitoring is not what monitoring tool we will use, but what we have to monitor and to what extent. In multi-cloud, monitoring must be cross-platform since we will have to see what's going on in the full chain of components that we have deployed in our multi-cloud environment. This is often referred to as end-to-end monitoring: looking at systems from an end user perspective. This is not only related to the health status of systems, but also whether the systems do what they should do and are free from bugs, crashes, or unexpected stops.
    - Monitoring systems collect logs. We will also need to design where these logs will have to go and how long these will have to be stored. The latter is important if systems are under an audit regime. Auditors can request logs.
    - Monitoring is also maybe even more related to the performance of these systems. From an end user perspective, there's nothing more frustrating than systems that respond slowly.
- **Performance**: Where an architect can decide that a system that responds within 1 second is fast, the end user might have a completely different definition of fast. Even if they agree that the performance and responsiveness of the system is slow, the next question is how to determine what the cause of degrading performance is. Monitoring the environment from the end user's perspective, all the way down to the infrastructure, is often referred to as end-to-end.

There are monitoring environments that really do end-to-end, typically by sending transactions through the whole chain and measuring health (heartbeat, to check if a system is still alive and responding) and performance by determining how fast transactions are processed. Keep in mind that this type of monitoring usually deploys agents on various components in your environment. In that case, we will have to take into consideration how much overhead these agents create on systems, thus taking up extra resources such as CPU and memory. The footprint of agents should be as small as possible, also given the fact that there will undoubtedly be more agents or packages running on our systems. Just think of endpoint protection, such as virus scanning, as an example.

- **Configurability**: Companies use cloud technology to be able to respond quickly to changes, and to become agile. As a result, systems are not static in the cloud. Systems in the cloud must be easy to configure. This is what configurability means: a set of parameters that define how systems must behave. Configurations set the desired state of a resource. As an example, we can use a virtual machine: developers can pick a VM from a portal of a cloud provider. This will be a default VM with default configurations. These configurations might not match the desired state for which the company has defined guidelines and guardrails. In that case, the specific desired configuration must be added separately. Configurations are defined in settings and configuration files. They might include:
    - CPU and memory allocation
    - Storage settings and disk usage
    - Boot sequence
    - Security settings such as hardening the system
    - Access management
    - Operating system configurations
- The configuration tells the resource how it should operate, and how it should "act." Our example is a VM, but it applies to every resource that we use in the cloud. Developers or administrators need to specify how a resource must be deployed, including all components that are part of the resource. Obviously, this is not something that developers and administrators want to do every time a specific resource is deployed. Configurations will have standards, and if there are standards, they can be automated. The moment a new resource is deployed, a process is automatically triggered to apply the desired settings to that resource. In that case, only the master configuration files have to be managed. Cloud providers offer tools to manage configurations, for instance, with **Desired State Configuration** (**DSC**), which is used in PowerShell.

DSC is declarative scripting that defines and automates the application of settings to Microsoft Windows and Linux systems. AWS offers Systems Manager for the same purpose. GCP has this embedded in Compute Engine.

- **Discoverability**: This is exactly what it says it is. It's about finding resources in cloud platforms. Resources must be visible for other resources in order to be able to communicate with each other and, obviously, these resources must be visible for monitoring. But there's more to discoverability. It also includes **service discovery**: the automatic detection of instances and services that run on these instances within a network, for instance, a tenant on a cloud platform:
    - This is extremely important in a microservices architecture: each microservice needs to know where instances are located to discover the services that are hosted on these instances. Service discovery allows the discovery of services dynamically without the need for IP addresses of the instances. These IP addresses will change dynamically, but the services will still be able to "find" each other. The technology typically involves a central services registry.
- **Security**: Here, we are focusing on data protection. After all, data is the most important asset in your multi-cloud environment. The architecture should have one goal: safeguarding the integrity of data. The best way to start thinking of security architecture is to think from the angle of possible threats. How do you protect the data, how do you prevent the application from being breached, and what are the processes in terms of security policies, monitoring, and following up on alerts? The latter is a subject for **Security Operations (SecOps)**. Security will be extensively discussed in part 4 of this book: *Controlling Security with DevSecOps.*
- **Scalability**: The killer feature of the public cloud is scalability. Whereas previously developers had to wait for the hardware to be ready in the data center before they could actually start their work, we now have the capacity right at our fingertips. But scalability goes beyond that: it's about full agility and flexibility in the public cloud and being able to scale out, scale up, and scale down whenever business requirements call for that.
- Scaling out is also referred to as horizontal scaling. When we scale out an environment, we usually add systems such as virtual machines to that environment. In scaling up—or vertical scaling—we add resources to a system, typically CPUs, memory, or disks in a storage system. Obviously, when we can scale out and up, we can also go the opposite direction and scale systems down. Since we are paying for what we really use in the public cloud, the costs will come down immediately, which is not the case if we have invested in physical machines sitting in a traditional, on-premises environment.

You will be able to lower the usage of these physical machines, but this will not lower the cost of that machine since it's fully **CAPEX—capital expenditures or investments**. We will discuss financials in part 3 of this book: *Controlling Costs in Multi-Cloud Using FinOps*.

- Typical domains for the scalability architecture are virtual machines (also as hosts for container clusters), databases, and storage, but network and security appliances should also be considered. For example, if the business demand for scaling their environment up or out increases, typically, the throughput also increases. This has an impact on network appliances such as switches and firewalls: they should scale too. However, you should use the native services from the cloud platforms to avoid scaling issues in the first place.
- Some things you should include in the architecture for scalability are as follows:
    - **Definition of scale units**: This concerns scalability patterns. One thing you have to realize is that scaling has an impact. Scaling out virtual machines has an impact on scaling the disks that these machines use, and thus the usage of storage.
    - But there's one more important aspect that an architect must take into account: can an application handle scaling? Or do we have to rearchitect the application so that the underlying resources can scale out, up, or down without impacting the functionality and, especially, the performance of the application? Is your backup solution aware of scaling?
    - Defining scale units is important. Scale units can be virtual machines, including memory and disks, database instances, storage accounts, and storage units, such as blobs in Azure or buckets in AWS. We must architect how these units scale and what the trigger is to start the scaling activity.
    - **Allowing for autoscaling**: One of the ground principles in the cloud is that we automate as much as we can. If we have done a proper job of defining our scale units, then the next step is to decide whether we allow autoscaling on these units or allow an automated process for dynamically adding or revoking resources to your environment. First, the application architecture must be able to support scaling in the first place. Autoscaling adds an extra dimension. The following aspects are important when it comes to autoscaling:
        - The trigger that executes the autoscaling process.
        - The thresholds of autoscaling, meaning to what level resources may be scaled up/out or down. Also, keep in mind that a business must have a very good insight into the related costs.

- **Partitioning**: Part of architecting for scalability is partitioning, especially in larger environments. By separating applications and data into partitions, controlling scaling and managing the scale sets becomes easier and prevents large environments from suffering from contention. Contention is an effect that can occur if application components use the same scaling technology but resources are limited due to set thresholds, which is often done to control costs.

Now, we have to make sure that our systems are not just scalable but are also resilient and robust, ensuring high availability:

- **Robustness**: Platforms such as Azure, AWS, and GCP are just there, ready to use. And since these platforms have global coverage, we can rest assured that the platforms will always be available. Well, these platforms absolutely have a high availability score, but they do suffer from outages. This is rare, but it does happen. The one question that a business must ask itself is whether it can live with that risk—or what the costs of mitigating that risk are, and whether the business is willing to invest in that mitigation. That's really a business decision at the highest level. It's all about business continuity.
- Robustness is about resilience, accessibility, retention, and recovery. In short, it's mostly about availability. When we architect for availability, we have to do so at different layers. The most common are the compute, application, and data layers. But it doesn't make sense to design availability for only one of these layers. If the virtual machines fail, the application and the database won't be accessible, meaning that they won't be available.
- In other words, you need to design availability from the top of the stack, from the application down to the infrastructure. If an application needs to have an availability of 99.9 percent, this means that the underlying infrastructure needs to be at a higher availability rate. The underlying infrastructure comprises the whole technology stack: compute, storage, and network.
- A good availability design counters for failures in each of these components, but also ensures the application—the top of the stack—can operate at the availability that has been agreed upon with the business and its end users. However, failures do occur, so we need to be able to recover systems. Recovery has two parameters:
    - **Recovery Point Objective (RPO)**: RPO is the maximum allowed time that data can be lost. An RPO could be, for instance, 1 hour of data loss. This means that the data that was processed within 1 hour since the start of the failure can't be restored. However, it's considered to be acceptable.
    - **Recovery Time Objective (RTO)**: RTO is the maximum duration of downtime that is accepted by the business.

- RPO and RTO are important when designing the backup, data retention, and recovery solution. If a business requires an RPO of a maximum of 1 hour, this means that we must take backups every hour. A technology that can be used for this is snapshotting or incremental backups. With snapshots, we take an instant copy of the data and save that copy, while incremental backups will take backups of the changes that have occurred since the last backup was made. Taking full backups every hour would create too much load on the system and, above that, implies that a business would need a lot of available backup storage.
- It is crucial that the business determines which environments are critical and need a heavy regime backup solution. Typically, data in such environments also needs to be stored for a longer period of time. Standard offerings in public clouds often have 2 weeks as a standard retention period for storing backup data. This can be extended, but it needs to be configured and you need to be aware that it will raise costs significantly.
- One more point that needs attention is that backing up data only makes sense if you are sure that you can restore it. So, make sure that your backup and restore procedures are tested frequently—even in the public cloud.
- **Portability:** There are multiple definitions of portability. Often, the term is referring to application portability. When the same application has to be installed on multiple platforms, portability can significantly reduce costs. The only requirement for portability is that there is a general abstraction between the application logic and the system interfaces. With cloud portability, we mean that applications can run on different cloud platforms: applications and data can be moved between clouds without (major) disruption. The topic is mostly related to the so-called "vendor" or "cloud" lock-in. Companies want to have the freedom to switch providers and suppliers, including cloud platforms. The term is also frequently mixed with interoperability, indicating that applications are cloud agnostic and can be hosted anywhere. Container technology using, for example, Kubernetes can run on any platform supporting Kubernetes.
- However, in practice, portability and interoperability are much harder than the theory claims. The underlying technology that cloud providers use might differ, causing issues in migrating from one cloud to another, especially with microservices that use specific native technology. One other aspect that plays an important role is data gravity, where the data literally attracts the applications toward the data. This can be a massive amount of data that can't be easily moved to another platform. Data gravity can heavily influence the cloud strategy of a company.

- **Usability**: This is often related to the ease of use of apps, with clear interfaces and transparent app navigation from a user's perspective. However, these topics do imply certain constraints on our architecture. Usability requires that the applications that are hosted in our multi-cloud environment are accessible to users. Consequently, we will have to think of how applications can or must be accessed. This is directly related to connectivity and routing: do users need access over the internet or are certain apps only accessible from the office network? Do we then need to design a **Demilitarized Zone (DMZ)** in our cloud network? And where are jump boxes positioned in multi-cloud?

    - Keep in mind that multi-cloud application components can originate from different platforms. Users should not be bothered by that: the underlying technical setup should be completely transparent for users. This also implies architectural decisions: something we call technology transparency. In essence, as architects, we constantly have to work from the business requirements down to the safe use of data and the secured accessibility of applications to the end users. This drives the architecture all the way through.

Continuous architecture is not the only framework that helps in defining agile architecture. While we mentioned TOGAF in this chapter, The Open Group, as an organization that develops standard methodologies for architecture, has issued a different framework that is more suitable for agile architecture than TOGAF, which in itself is rather static. For all good reasons, this new framework is called **Open Agile Architecture** (**O-AA**).

The framework addresses business agility as the main purpose for doing architecture in enterprises, embracing the digital transformation these enterprises are going through. O-AA puts the customer and the customer experience at the heart of the architecture. Since customer demands and experiences change continuously, the architecture must be able to adopt this: it catches the interactions between the customer and all the touchpoints the customer has with the enterprise, for example, through the products of the enterprise, but also the website, social media, customer contact centers and even the **Internet of Things (IoT)** such as mobile devices and sensors that are connected to the internet.

This book will not provide deep dives into TOGAF or O-AA. Please refer to the website of The Open Group for more information and the collateral. O-AA was released in 2020. Van Haren Publishing recently released the methodology in book form, available from TOGAF at `https://publications.opengroup.org/`

Next to the quality attributes, we also must define some principles for our multi-cloud architecture. This is discussed in the next section.

Defining principles from use cases

Before we start defining the architecture, we must define the business use case.

A business use case describes the actions a business must take in a specific, defined order to deliver a product or a service to a customer. That product or service must represent a value to that customer since that value can be represented by a price. That price in turn must match the costs that are related to the delivery of the product or service, topped by a margin so that the business makes a profit. So, the business use case describes the workflow to produce and deliver but also defines how the business adds value for its customers while making a profit.

From the use case, the architect derives the principles, starting with the business principles.

Business principles

Business principles start with business units setting out their goals and strategy. These adhere to the business mission statement and, from there, describe what they want to achieve in the short and long term. This can involve a wide variety of topics:

- Faster response to customers
- Faster deployment of new products (time to market)
- Improve the quality of services or products
- Engage more with employees
- Real digital goals such as releasing a new website or web shop
- Increase revenue and profit, while controlling costs

The last point, controlling costs, is a topic of FinOps, which will be discussed in chapters 10, *Managing Costs with FinOps*, and 11, *Improving Cost Management with the FinOps Maturity Model*.

As with nearly everything, goals should be **SMART**, which is short for **specific, measurable, attainable, relevant, and timely**. For example, a SMART-formulated goal could be "*the release of the web shop for product X in the North America region on June 1*." It's scoped to a specific product in a defined region and targeted at a fixed date. This is measurable as a SMART goal.

Coming back to TOGAF, this is an activity that is performed in phase B of the ADM cycle; that is, the phase in the architecture development method where we complete the business architecture. Again, this book is not about TOGAF, but we do recommend having one language to execute the enterprise architecture in. TOGAF is generally seen as the standard in this case. Business principles drive the business goals and strategic decisions that a business makes. For that reason, these principles are a prerequisite for any subsequent architectural stage.

Principles for security and compliance

Though security and compliance are major topics in any architecture, the principles in this domain can be fairly simple. Since these principles are of extreme importance in literally every single aspect of the architecture, it's listed as the second most important group of principles, right after business principles.

Nowadays, we talk a lot about zero trust and security by design. These can be principles, but what do they mean? **Zero trust** speaks for itself: organizations that comply with zero trust do not trust anything within their networks and platforms. Every device, application, or user is monitored. Platforms are micro-segmented to avoid devices, applications, and users from being anywhere on the platform or inside the networks: they are strictly contained. The pitfall here is to think that zero trust is about technological measures only. It's not. Zero trust is first and foremost a business principle and looks at security from a different angle: zero trust assumes that an organization has been attacked, with the only question left being what the exact damage was. This is also the angle that frameworks such as MITRE ATT&CK take.

Security by design means that every component in the environment is designed to be secure from the architecture of that component: built-in security. This means that platforms and systems, including network devices, are hardened and that programming code is protected against breaches via encryption or hashing. This also means that the architecture itself is already micro-segmented and that security frameworks have been applied. An example of a commonly used framework is the **Center for Internet Security (CIS)** framework, which contains 20 critical security controls that cover various sorts of attacks on different layers in the IT stack. As CIS themselves rightfully state, it's not a one size fits all framework. An organization needs to analyze what controls should be implemented and to what extent.

We'll pick just one as an example: data protection, which is control 13 in the CIS framework. The control advises that data in transit and data at rest are encrypted. Note that CIS doesn't say what type of **Hardware Security Modules (HSMs)** an organization should use or even what level of encryption.

It says that an organization should use encryption and secure this with safely kept encryption keys. It's up to the architect to decide on what level and what type of encryption should be used.

In terms of compliance principles, it must be clear what international, national, or even regional laws and industry regulations the business has to adhere to. This includes laws and regulations in terms of privacy, which is directly related to the storage and usage of (personal) data.

An example of a principle is that the architecture must comply with the **General Data Protection Regulation (GDPR)**. This principle may look simple—comply with GDPR—but it means a lot of work when it comes to securing and protecting environments where data is stored (the systems of record) and how this data is accessed (systems of engagement). Technical measures that will result from this principle will vary from securing databases, encrypting data, and controlling access to that data with authentication and authorization. In multi-cloud, this can be even more challenging than it already was in the traditional data center. By using different clouds and PaaS and SaaS solutions, your data can be placed anywhere in terms of storage and data usage.

Data principles

As we mentioned previously, here's where it really gets exciting and challenging at the same time in multi-cloud environments. The most often used data principles are related to data confidentiality and, from that, protecting data. The five most important data principles are:

- **Accuracy**: Data should be accurate, complete, and reliable. Inaccurate data can lead to flawed insights and decisions and can have serious consequences.
- **Relevance**: Data should be relevant to the problem or question at hand. Unnecessary or irrelevant data can add noise to the analysis and make it harder to extract meaningful insights.
- **Timeliness**: Data should be timely and up to date. Outdated or stale data can lead to incorrect or misleading conclusions.
- **Consistency**: Data should be consistent in terms of format, definitions, and units of measurement. Inconsistent data can create confusion and make it difficult to combine or compare different sources of information.
- **Privacy and security**: Data should be protected from unauthorized access and use. Sensitive data, such as personal or financial information, should be handled with care and stored securely to avoid breaches or leaks.

So, how do you ensure that these principles are adhered to? We briefly touched on two important technology terms that have become quite common in cloud environments earlier in this chapter:

- **Systems of record**: Systems of record are data management or information storage systems; that is, systems that hold data. In the cloud, we have the commonly known database, but due to the scalability of cloud platforms, we can now deploy huge data stores comprising multiple databases that connect thousands of data sources. Public clouds are very suitable to host so-called data lakes.
- **Systems of engagement**: Systems of engagement are systems that are used to collect or access data. This can include a variety of systems: think of email, collaboration platforms, and content management systems, but also mobile apps or even IoT devices that collect data, send it to a central data platform, and retrieve data from that platform.

A high-level overview of the topology for holding systems of record and systems of engagement is shown in the following diagram, with **Enterprise Resource Planning (ERP)**, **Content Management (CMS)**, and **Customer Relationship Management (CRM)** systems being used as examples of systems of record:

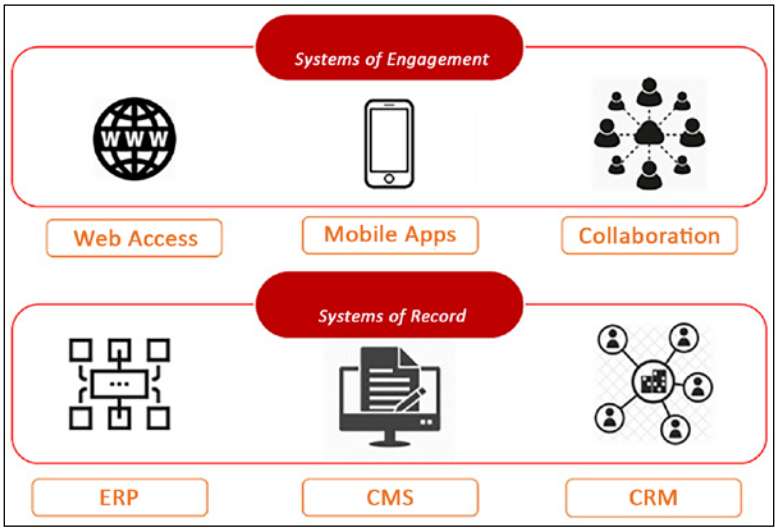

Figure 5.2: Simple representation of systems of engagement and systems of record

The ecosystem of record and engagement is enormous and growing. We've already mentioned data lakes, which are large data stores that mostly hold raw data. In order to work with that data, a data scientist would need to define precise datasets to perform analytics. Azure, AWS, and Google all have offerings to enable this, such as Data Factory and Databricks in Azure, EMR and Athena in AWS, and BigQuery from Google.

Big data and data analytics have become increasingly important for businesses in their journey to become data-driven: any activity or business decision, for that matter, is driven by actual data. Since clouds can hold petabytes of data and systems need to be able to analyze this data fast to trigger these actions, a growing number of architects believe that there will be a new layer in the model. That layer will hold "systems of intelligence" using machine learning and **artificial intelligence (AI)**. Azure, AWS, and Google all offer AI-driven solutions, such as Azure ML in Azure, SageMaker in AWS, and Cloud AI in Google. The extra layer—the systems of intelligence—can be seen in the following diagram:

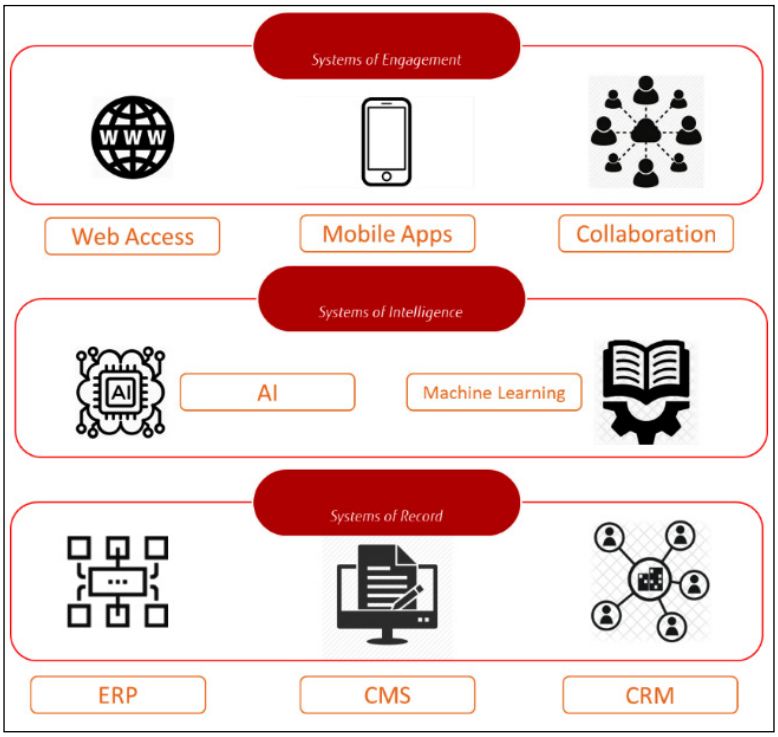

Figure 5.3: Simple representation of the systems of intelligence layer

To be clear: systems of record or engagement don't say anything about the type of underlying resource. It can be anything from a physical server to a **virtual machine (VM)**, a container, or even a function. Systems of record or engagement only say something about the functionality of a specific resource.

Application principles

Data doesn't stand on its own. If we look at TOGAF once more, we'll see that data and applications are grouped into one architectural phase, known as phase C, which is the information systems architecture. In modern applications, one of the main principles of applications is that they have a data-driven approach, following the recommendation of Steven Spewak's enterprise architecture planning. Spewak published his book *Enterprise Architecture Planning* in 1992, but his approach is still very relevant, even—and perhaps even more—in multi-cloud environments.

Also mentioned in Spewak's work: the business mission is the most important driver in any architecture. That mission is data-driven; enterprises make decisions based on data, and for that reason, data needs to be relevant, but also accessed and usable. These latter principles are related to the applications disclosing the data to the business. In other words, applications need to safeguard the quality of the data, make data accessible, and ensure that data can be used. Of course, there can be—and there is—a lot of debate regarding, for instance, the accessibility of data. The sole relevant principle for an application architecture is that it makes data accessible. To whom and on what conditions are both security principles.

In multi-cloud, the storage data changes, but also the format of applications. Modern applications are usually not monolithic or client-server-based these days, although enterprises can still have a large base of applications with legacy architectures. Cloud-native apps are defined with roles and functions and build on the principles of code-based modularity and the use of microservices. These apps communicate with other apps using APIs or even triggers that call specific functions in other apps. These apps don't even have to run on the same platform; they can be hosted anywhere. Some architects tend to think that monolithic applications on mainframes are complex, so use that as a guideline to figure out how complex apps in multi-cloud can get.

However, a lot of architectural principles for applications are as valid as ever. The technology might change, but the functionality of an application is still to support businesses when it comes to rendering data, making it accessible, and ensuring that the data is usable.

Today, popular principles for applications are taking the specific characteristics of cloud-native technology into consideration. Modern apps should be enabled for mobility, be platform-independent using open standards, support interoperability, and be scalable. Apps should enable users to work with them at any time, anywhere.

One crucial topic is the fact that the requirements for applications change at almost the speed of light: users demand more and more from apps, so they have to be designed in an extremely agile way so that they can adopt changes incredibly fast in development pipelines. Cloud technology does support this: code can easily be adapted. But this does require that the applications are well-designed and documented, including in runbooks.

Infrastructure and technology principles

Finally, we get to the real technology: machines, wires, nuts, and bolts. Here, we're talking about virtual nuts and bolts. Since data is stored in many places in our multi-cloud environment and applications are built to be cloud-native, the underlying infrastructure needs to support this. This is phase D in TOGAF, the phase in architecture development where we create the target technology architecture, which comprises the platform's location, the network topology, the infrastructure components that we will be using for specific applications and data stores, and the system interdependencies. In multi-cloud, this starts with drafting the landing zone: the platform where our applications and data will land.

One of the pitfalls of this is that architects create long, extensive lists with principles that infrastructure and technology should adhere to, all the way up to defining the products that will be used as a technology standard. However, a catalog with products is part of a portfolio. Principles should be generic and guiding, not constraining. In other words, a list of technology standards and products is not a principle. A general principle could be about bleeding edge technology: a new, non-proven, experimental technology that imposes a risk when deployed in an environment because it's still unstable and unreliable.

Other important principles for infrastructure can be that it should be scalable (scale out, up, and down) and that it must allow micro-segmentation. We've already talked about the Twelve-Factor App, which sets out specific requirements for the infrastructure. These can be used as principles. The principles for the Twelve-Factor App were set out in 2005, but as we already concluded in *Chapter 2*, *Collecting Business Requirements*, they are still very accurate and relevant.

The Twelve-Factor App sets the following three major requirements for infrastructure:

- The app is portable between different platforms, meaning that the app is platform-agnostic and does not rely on a specific server or system's settings.
- There's little to no difference between the development stage and the production stage of the app so continuous development and deployment are enabled. The platform that the app is deployed on should support this (meaning that everything is basically code-based).

- The app supports scaling up without significant changes needing to be made to the architecture of the app.

In the next section, we will discuss the principles for usability and processes. We will also touch upon the transition and transformation to cloud environments.

Principles for processes

The last group of principles is concerned with processes. This is not about the **IT System Management (ITSM)** processes, but about the processes of deployment and automation in multi-cloud. One of the principles in multi-cloud is that we will automate as much as we can. This means that we will have to define all the tasks that we would typically do manually in an automated workflow. If we have a code-only principle defined, then we can subsequently set a principle that states that we must work from the code base or master branch. If we fork the code and we do have to make changes to it, then a principle is that altered code can only be committed back to the master code if it's tested in an environment that is completely separated from acceptance and production. This is related to the life cycle process of our environment.

So, processes here focus more on our way of working. Today, a lot of companies are devoted to agile and DevOps. If that's the defined way of working, then it should be listed as a principle; for example, an organization embraces agility in its core processes. This can be done through the **Scaled Agile Framework (SAFe)** or the Spotify model. Following that principle, a company should also define the teams, their work packages, and how epics, features, product backlogs, and so on are planned. However, that's not part of the principle anymore. That's a consequence of the principle.

As with all principles, the biggest pitfall is making principles too complex. Especially with processes, it's important to really stick to describing the principle and not the actual process.

We have discussed the architecture principles and why we are doing architecture. The next step is to define the components of the architecture. This is the topic of the next section.

Creating the architecture artifacts

The hierarchy in documents that cover the architecture starts with the enterprise architecture. It's the first so-called architecture artifact. The enterprise architecture is followed by the high-level design and the low-level design, which covers the various components in the IT landscape. We will explore this in more detail in the following sections. Keep in mind that these sections are merely an introduction to the creation of these artifacts. You will find samples of these artifacts at `https://publications.opengroup.org/i093`, where you can download a ZIP file containing relevant templates.