**4**

**Service Designs for Multi-Cloud**

All cloud providers offer a cloud adoption framework that helps businesses to implement governance and deploy services, while controlling service levels and **key performance indicators** (**KPIs**) in the cloud. In multi-cloud environments, businesses would have to think about how to implement governance over different cloud components, coming from different providers, while still being able to manage it as a single environment.

This chapter will introduce the base pillars for a unified service design and governance model, starting with identities. Everything in the cloud is an identity; anything that has a role and therefore needs access to resources is an identity. It requires a different way of thinking—users, VM, a piece of code even. We will look at the different pillars of multi-cloud governance and study the various stages in cloud adoption, using the Well-Architected and cloud adoption frameworks of cloud providers.

In this chapter, we will cover the following topics:

* Introducing the scaffold for multi-cloud environments
* Working with Well-Architected Frameworks
* Understanding cloud adoption
* Translating business KPIs into cloud SLAs
* Using cloud adoption frameworks to align between cloud providers
* Understanding identities and roles in the cloud
* Creating the service design and governance model

**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.

The frameworks typically contain five pillars, or scaffolds. The frameworks for AWS, Azure, and GCP all provide guidelines for:

* Operational excellence
* Security
* Cost optimization
* Reliability
* Performance efficiency
* Sustainability (at the time of writing, only included by AWS)

Architects can now use these frameworks to design solutions in the cloud. The Well-Architected Frameworks will show how to optimize workloads in the cloud, with control of costs and optimized serviceability. It also contains guardrails to properly secure workloads to keep data in the cloud protected. The frameworks help in implementing **financial controls** (**FinOps**), **security controls** (**SecOps**), and **management of workloads** (**BaseOps**). All of these will be extensively discussed during the course of this book.

The Well-Architected Frameworks offer best practices to validate cloud architectures. These frameworks contain the following principles, guidelines, and best practices.

First, Well-Architected Frameworks offer guidelines to run workloads in the most efficient way, using the best solution in a specific cloud. Solutions can be VMs, but also containers and serverless options. **Operational excellence** will guide us in setting up efficient operations and management of cloud environments, for instance, by using automation and concepts such as **Infrastructure as Code** (**IaC**).

We must keep our environments secure and compliant. Well-Architected Frameworks provide best practices to manage the security posture of workloads that are hosted in the cloud. Remember that the cloud provider is responsible for the cloud, but the customer is responsible for what is in the cloud. AWS, Azure, GCP, and other providers offer a variety of tools to secure workloads and data, but it’s up to the customer to use these tools. The most important question to be answered is: *who is allowed to do what, when, and why?*This is addressed in implementing **Identity and Access Management** (**IAM**).

Managing security also includes compliancy. Companies often need to be compliant with industry standards and regulations. Well-Architected Frameworks will guide you in setting up compliance with the most common-known compliance frameworks, such as PCI-DSS, NIST, and ISO.

**Cost management**: Companies must know what they spend in the cloud and if it’s adding value to the business. Cost management, or **FinOps**, monitors cloud spend, but also offers tools to control costs with, for instance, budgeting, alerting, and rightsizing workloads. Rightsizing may involve the choice of serverless solutions instead of VMs, or using reserved instances when workloads are needed for a longer period. With reserved instances, companies can save up to 70 percent of the VM cost compared to pay as you go.

**Establish reliability**: The cloud provides solutions that ensure high availability by spreading workloads over various zones in the cloud data centers and even (global) regions. But reliability is also about automation, for instance, through automatic failovers, recovery, and scaling solutions to cover peak loads.

As mentioned, sustainability is a relatively new pillar in frameworks. This pillar focuses on using solutions that require less energy and making efficient use of resources in the cloud in order to reduce energy consumption by optimizing the utilization of power and cooling. This topic is covered in *Chapter 19*, *Introducing AIOps and GreenOps*,where we will introduce and discuss GreenOps in more detail.

In the next sections, we will elaborate a bit more on topics that must be addressed regarding the Well-Architected Frameworks.

**Identity and Access Management (IAM)**

Who may do what, when, and why? It is key to understand this in the public cloud, so we will devote more words to identity and access later on in this chapter, in the section *Understanding identities and roles in the cloud*. The most important thing to bear in mind is that virtually everything in the cloud is an identity. We are not only talking about people here, but also about resources, functions, APIs, machines, workloads, and databases that are allowed to perform certain actions. All these resources need to be uniquely identified in order to authenticate them in your environment.

Next, specific access rules need to be set for these identities; they need to be authorized to execute tasks. Obviously, identity directory systems such as Active Directory or OpenLDAP are important as identity providers or identity stores. The question is whether you want this store in your public cloud environment or whether you want an authentication and authorization mechanism in your environment, communicating with the identity store. As everything is transforming into an “as a service” model, architects could also choose identity as a service. An example of this is Okta, which manages identity and access across multiple clouds.

As stated previously, we will look into this in more detail later on in this chapter under the heading *Understanding identities and roles in the cloud*, since this is the key starting point in multi-cloud architecture and governance.

**Security**

Here’s a bold statement: platforms such as Azure, AWS, and GCP are likely the most secure platforms in the world. They have to be, since thousands of companies host their systems on these platforms.

Yet, security remains the responsibility of the business itself. Cloud platforms will provide you with the tools to secure your environment. Whether you want to use these tools and to what extent is entirely down to the business. Security starts with policies. Typically, businesses will have to adhere to certain frameworks that come with recommendations or even obligations to secure systems. As well as these industry standards, there are horizontal security baselines, such as the **Center for Internet Security** (**CIS**). The CIS baseline is extensive and covers a lot of ground in terms of hardening resources in the cloud. The baseline has scored and non-scored items, where the scored items are obviously the most important. Auditors will raise an issue on these items when the baseline is not met.

One framework deserves a bit more attention: MITRE ATT&CK. Having a baseline implemented is fine, but how do you retain control over security without being aware of actual attack methods and exploited vulnerabilities? MITRE ATT&CK is a knowledge base that is constantly evaluated with real-world observations in terms of security attacks and known breaches. The keyword here is real-world. They keep track of real attacks, and mitigation of these attacks, for the major public clouds—Azure, AWS, and GCP—but also for platforms that are enrolled on top of these platforms, such as Kubernetes for container orchestration.

The MITRE ATT&CK matrices can be found at <https://attack.mitre.org/>.

There’s a lot to say about securing your cloud platform. *Chapter 16*, *Defining Security Policies for Data*, provides best practices in relation to cloud security.

**Cost management**

You’ve likely heard this before from companies that enter the cloud: “the public cloud is not as cheap as the traditional stack that I run in my own data center.” That could be true. If a business decides to perform a lift and shift from traditional workloads to the public cloud, without changing any parameters, they will realize that hosting that workload 24/7/365 in the public cloud is probably more expensive than having it in an on-premises system. There are two explanations for that:

* Businesses have a tendency not to fully calculate all related costs to workloads that are hosted on on-premises systems. Very often, things such as power, cooling, but also labor (especially involved in changes) are not taken into consideration.
* Businesses use the public cloud like they use on-premises systems, but without the functionality that clouds offer in terms of flexibility. Not all workloads need to be operational 24/7/365. Cloud systems offer a true pay-as-you-go model, where businesses only pay for the actual resources they consume—even all the way down to the level of CPUs, memory, storage, and network bandwidth.

Cost management is of the utmost importance. You want to be in full control of whatever your business is consuming in the cloud. But that goes hand in hand with IAM. If a worker has full access to platforms and, for instance, enrolls a heavy storage environment for huge data platforms such as data lakes, the business will definitely get a bill for that. There are two possible problems here:

* First of all, was that worker authorized to perform that action?
* Second, has the business allocated a budget for that environment?

Let’s assume that the worker was authorized, and that budget is available. Then we need to be able to track and trace the actual consumption and, if required, be able to put a chargeback in place for a certain division that uses the environment. Hence, we need to be able to identify the resources. That’s where naming and tagging in resources come into play. We have to make sure that all resources can uniquely be identified by name. Tags are used to provide information about a specific resource, for example, who owns the resource and hence will have to pay to use that resource.

*Chapter 12*, *Cost and Value Modeling in Multi-Cloud Propositions*, provides best practices on how to implement FinOps.

**Monitoring**

We need visibility on and in our platform. What happens and for what reason? Is our environment still healthy in terms of performance and security? Cloud providers offer native monitoring platforms: Azure Monitor, AWS CloudWatch, Google Cloud Monitoring (previously known as Stackdriver), and CloudMonitor in Alibaba Cloud. The basic functionality is comparable all in all: monitoring agents collect data (logs) and metrics, and then store these in a log analytics space where they can be explored using alert policies and visualization with dashboards. All these monitoring suites can operate within the cloud itself, but also have APIs to major **IT Service Management** (**ITSM**) systems such as ServiceNow, where the latter will then provide a single pane of glass over different platforms in the IT environment.

There are quite a number of alternatives to the native toolsets and suites. Good examples of such tools are Splunk and Datadog. Some of these suites operate in the domain of **AIOps**, a rather new phenomenon. AIOps does a lot more than just execute monitoring: it implies intelligent analyses conducted on metrics and logs. In *Chapter 19*, *Introducing AIOps and GreenOps*, we will cover AIOps, since it is expected that this will become increasingly popular in multi-cloud environments.

**Automation**

We have already talked about costs. By far the most expensive factor in IT is costs related to labor. For a lot of companies, this was the driver to start offshoring labor when they outsourced their IT. But we have already noticed that IT has become a core business again for these companies. For example, a bank is an IT company these days. They almost completely outsourced their IT in the late nineties and at the beginning of the new millennium, but we now see that their IT functions are back in-house. However, there is still the driver to keep costs down as much as possible. With concepts such as Infrastructure as Code and Configuration as Code, repositories to store that code, and having the deployment of code fully automated using DevOps pipelines, these companies try to automate to the max. By doing this, the amount of labor is kept to a minimum.

Costs are, however, not the only reason to automate, although it must be said that this is a bit of an ambiguous argument. Automation tends to be less fault tolerant than human labor (*robots do it better*). Most large outages are caused by human error that could have been avoided by automation. However, automation only works when it’s tested thoroughly. And you still need good developers to write the initial code and the automation scripts. True, the cloud does offer a lot of good automation tools to make the lives of businesses easier, but we will come to realize that automation is not only about tools and technology.

It’s also about processes; methodologies such as **Site Reliability Engineering** (**SRE**)—invented by Google—are gaining a lot of ground.

These pillars are described in almost all frameworks. In the following sections, we will study these pillars as we go along the various stages involved in cloud adoption.

**Understanding cloud adoption**

You may have come across one other term: the **cloud landing zone**. The landing zone is the foundation environment where workloads eventually will be hosted. Picture it all like a house. We have a foundation with a number of pillars. On top of that, we have a house with a front door (be aware that this house should not have a back door), a hallway, and a number of rooms. These rooms will be empty: no decoration, no furniture. That all has yet to be designed and implemented, where we have some huge shops (portals) from where we can choose all sorts of solutions to get our rooms exactly how we want them. The last thing to do is to actually move the residents into the house. And indeed, these residents will likely move from room to room. Remember: without scaffolding, it’s hard to build a house in the first place.

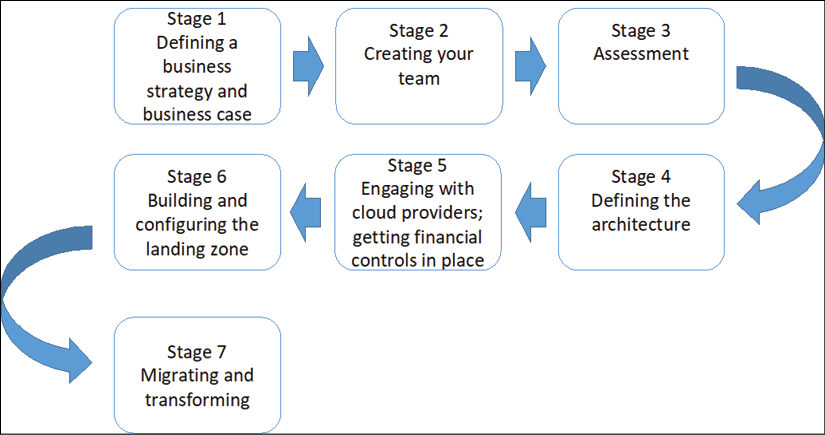
This is what cloud adoption frameworks are all about: how to adopt cloud technology.

The respective cloud adoption frameworks can be found as follows:

* The Azure Cloud Adoption Framework: <https://azure.microsoft.com/en-us/cloud-adoption-framework/#cloud-adoption-journey>
* AWS: <https://aws.amazon.com/professional-services/CAF/>
* GCP: <https://cloud.google.com/adoption-framework/>

Often, this is referred to as a journey. Adoption is defined by a number of stages, regardless of the target cloud platform. Governing and managing are cross-processes supporting those stages.

The following diagram shows the subsequent stages of cloud adoption:



*Figure 4.1: The seven steps in cloud adoption*

Let’s discuss each stage in detail in the following sections.

**Stage 1—Defining a business strategy and business case**

In *Chapter 2*, *Business Acceleration Using a Multi-Cloud Strategy*, we looked extensively at the business strategy and how to create a business case. A business needs to have clear goals and know where cloud offerings could add value. We have also looked at technical strategies such as rehost, replatform, and rebuild. Rebuild might not always be the cheapest solution when it comes to **total cost of ownership** (**TCO**). That’s because businesses tend to forget that it takes quite some effort to rearchitect and rebuild applications to cloud-native environments. This architecture and build costs should be taken into consideration. However, the native environment will undoubtedly bring business benefits in terms of flexibility and agility. In short, this first stage is crucial in the entire adoption cycle.

**Stage 2—Creating your team**

Let’s break this one immediately: there’s no such thing as a T-shaped professional, someone who can do everything in the cloud, from coding applications to configuring infrastructure. The wider the T, the less deep the stroke under the T gets. In other words, if you have professionals who are generic and basically have a knowledge of all cloud platforms, they will likely not have a deep understanding of one specific cloud technology. But also, a software developer is not a network engineer, and vice versa.

Of course, someone who’s trained in architecting and building cloud environments in Azure or AWS will probably know how to deploy workloads and a database and have a basic understanding of connectivity. But what about firewall rules? Or specific routing? Put another way: a firewall specialist might not be very skilled at coding in Python. If you have these types of people on your team, congrats. Pay them well, so they won’t leave. But you will probably have a team with mixed skills. You will need developers and staff that are proficiently trained in designing and configuring infrastructure, even in the cloud.

Some adoption frameworks refer to this as the cloud center of excellence or—as AWS calls it—the Cloud Adoption Office, which is where the team with all the required skills is brought together. Forming this center of excellence is an important step in adopting cloud technology.

**Stage 3—Assessment**

The assessment phase is a vital step in ensuring proper migration to a target cloud environment. Before we start migrating or rebuilding applications in our landing zone, we need to know what we have. This is also important for the landing zone that we are planning to build. Look at it this way: you need to know what the house will look like before you start laying the foundation of that house. You need a blueprint.

First and foremost, we need to assess our business strategy. Where do we want to go with our business and what does it take to get there? The next question would be: is our current IT state ready to deliver that strategy? Which applications do we have and what business function do they serve? Are applications and the underlying infrastructure up to date or are they (near) the end of service, or even the end of life? What support contracts do we have and do we need to extend these during our transformation to the cloud, or can we retire these contracts?

You get the point: a proper assessment takes some time, but don’t skip it. Our end goal should be clear; at the end of the day, all businesses want to become digital companies steered by data-driven decisions. We need an environment in which we can disclose data in a secure way and make the environment as flexible and scalable as possible so it can breathe at business speed. The problem is our traditional IT environment, which has been built out over many years. If we’re lucky, everything will be well documented, but the hard reality is that documentation is rarely up to date. If we start our transformation without a proper assessment, we are bound to start pulling bricks out of an old wall without knowing how stable that wall really is. Again, here too, we require scaffolds.

**Stage 4—Defining the architecture**

This is the stage where you will define the **landing zone**—the foundation platform where the workloads will be hosted, based on the outcomes of the assessment. It starts with connectivity so that administrators and developers can reach the workloads in the cloud. Typically, connections will terminate in a transit zone or hub, the central place where inbound and outbound traffic is regulated and filtered by means of firewalls, proxies, and gateways.

Most administrators will access systems through APIs or consoles, but in some cases, it might be recommended to use a jump server, a stepping stone, or a bastion server, the server that forms the entrance to the environment before they can access any other servers. Typically, third parties such as system integrators use this type of server. In short, this transit zone or hub is crucial to the architecture.

The next thing is to define the architecture according to your business strategy. That defines how your cloud environment is set up. Does your business have divisions or product lines? It might be worthwhile having the cloud environment corresponding to the business layout, for instance, by using different subscriptions or the **Virtual Private Cloud** (**VPC**) per division or product line.

There will be applications that are more generically used throughout the entire business. Office applications are usually a good example. Will these applications be hosted in one separate subscription? And what about access for administrators? Does each of the divisions have its own admins controlling the workloads? Has the business adopted an agile way of working, or is there one centralized IT department that handles all of the infrastructure? Who’s in charge of the security policies? These policies might differ by division or even workload groups. These security policies might also not be cloud friendly or applicable in a cloud-native world. They might need to be updated based on feedback from your cloud **Subject Matter Experts** (**SMEs**), the specialists.

One major distinction that we can already make is the difference between systems of record and systems of engagement, both terms first used by Microsoft. Systems of record are typically backend systems, holding the data. Systems of engagement are the frontend systems used to access data, work with the data, and communicate said data. We often find this setup reflected in the tiering of environments, where tier 1 is the access layer, tier 2 is the worker (middleware), and tier 3 is the database layer. A common rule in architecture is that the database should be close to the application accessing the database. In the cloud, this might work out differently, since we will probably work with **Platform as a Service** (**PaaS**) as a database engine.

These are the types of questions that are addressed in the cloud adoption frameworks. They are all very relevant questions that require answers before we start. And it’s all architecture. It’s about mapping the business to a comprehensive cloud blueprint. *Chapter 5*,*Managing the Enterprise Cloud Architecture*, of this book is all about architecture.

**Stage 5—Engaging with cloud providers; getting financial controls in place**

In this stage, we will decide on which cloud platform we will build the environment and what type of solutions we will be using: **Infrastructure as a Service** (**IaaS**), PaaS, **Software as a Service** (**SaaS**), containers, or serverless. This solution should be derived from the architecture that we have defined in stage 3. We will have to make some make-or-buy decisions: can we use native solutions, off the shelf, or do we need to develop something customized to the business requirements?

During this stage, we will also have to define business cases that automatically come with make-or-buy analyses. For instance, if we plan to deploy VMs on IaaS, we will have to think of the life cycle of that VM. In the case of a VM that is foreseen to live longer than, let’s say for the sake of argument, 1 year, it will be far more cost efficient to host it on reserved instances as opposed to using the pay-as-you-go deployment model. Cloud providers offer quite some discounts on reserved instances, for a good reason: reserved instances mean a long-term commitment and, hence, a guaranteed income. But be aware: it’s a long-term commitment. Breaking that commitment comes at a cost. Do involve your financial controller to have it worked out properly.

Development environments will generally only exist for a shorter time. Still, cloud providers do want businesses to develop as much as possible on their cloud platforms and offer special licenses for developers that can be really interesting. At the end of the day, cloud providers are only interested in one thing: the consumption of their platforms. There are a lot of programs that offer all sorts of guidance, tools, and migration methods to get workloads to these platforms.

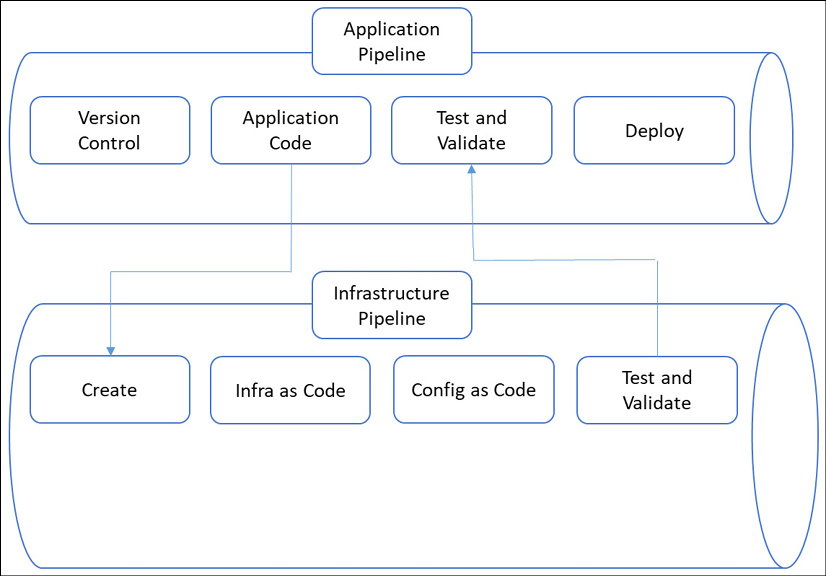
**Stage 6—Building and configuring the landing zone**

There are a number of options to actually start building the landing zone. Just to ensure we understand: the landing zone is the foundation platform, typically the transit zone or hub and where the basic configuration of VNets, VPCs, or projects is carried out. We aim to have it automated as much as we can, right from the start. Hence, we will work according to the Infrastructure as Code and Configuration as Code, since we can only automate when components are code-based.

However, there are, of course, other ways to start your build, for instance, using the portals of the respective cloud providers.

If you are building a small, rather simple environment with just a few workloads, then the portal is the perfect way to go. But assuming that we are working with an enterprise, the portal is not a good idea to build your cloud environment.

It’s absolutely fine to start exploring the cloud platform, but as the enterprise moves along and the environments grow, we need a more flexible way of managing and automating workloads. As has already been said, we want to automate as much as we can. How do we do that? By coding our foundation infrastructure and defining that as our master code. That master code is stored in a repository. Now, from that repository, we can fork the code if we need to deploy infrastructure components. It is very likely that every now and then, we have to change the code due to certain business requirements. That’s fine, as long as we merge the changed, approved code back into the master repository. By working in this way, we have deployed an infrastructure pipeline, shown as follows:



*Figure 4.2: Basic pipeline structure*

In a multi-cloud environment, the biggest challenge would be having a consistent repository and pipeline that could expand over multiple cloud platforms. After all, although the base concepts of AWS, Azure, and GCP are more or less the same, they do differ in terms of applied technology.

In this section, we are mainly talking about IaaS components. A popular tool in this area is Terraform, an open-source software by HashiCorp that is specifically designed to deploy data center infrastructure. Terraform abstracts the code of these clouds into cloud-agnostic code, based on HCL or JSON. **HashiCorp Configuration Language** (**HCL**) is the native language of Terraform, while **JavaScript Object Notation** (**JSON**) is more commonly known as a markup language.

If you search for alternatives, you may find tools such as Ansible, Chef, Puppet, or SaltStack. However, these are configuration tools and work quite differently from a provisioning tool such as Terraform.

**Stage 7—Migrating and transforming**

Now we have our landing zone and we’re ready to start deploying services and workloads into our cloud environments. This is what we call **cloud transformation**. This is the stage where we will implement the technical strategies that we discussed in *Chapter 2*, *Business Acceleration Using a Multi-Cloud Strategy*, such as rehost, replatform, rebuild, retain, and retire. Following the assessment, we have defined for each application or application group what the business focus is and what the technical strategy is. In this final stage, we will shift our applications to the new platform, apply new cloud services, or rebuild applications native to the cloud. Obviously, we will do that in a DevOps way of working, where we do all the building and configuration from code, as we discussed in stage 5.

DevOps typically comprises the agile concepts of epics and features. Don’t overcomplicate things. The epic could be *implementing new cloud architecture* and the transformation of an application or application group can be the feature. With the knowledge that we acquired from the assessment phase, combined with the technical strategy, the DevOps team should be able to do a good refinement, breaking down the feature into tasks that can be executed and completed in a limited number of sprints.

There are two really important items in a transformation stage: going live and the exit strategy. Before going live, testing is required. The need for good testing is really underestimated: it must be part of the DevOps CI/CD pipeline. It is very strongly advised to run the full test cycle: unit, integration, and end user test. From a process point of view, no company should allow anything to go live before all test findings have proven to be addressed. The same applies to the exit strategy: no company should allow anything to go live without a clearly defined exit strategy, that is, how to roll back or move environments out of the cloud again, back to the original state.

It’s one more reason to consider rebuilding environments, so that there’s always the old environment as a fallback if things turn out not to be working as designed. Of course, testing should prevent things from going wrong, but we have to be realistic too: something will break.

**Translating business KPIs into cloud SLAs**

Frankly, infrastructure in the cloud should be a black box for a business. Infrastructure is like turning on a water tap. Some IT companies referred to operating cloud infrastructure as liquid or fluid IT for that reason: it was simply there, all the time. As a consequence, the focus on SLAs shifted to the business itself. Also, that is part of cloud adoption. As enterprises are moving ahead in the adoption process, a lot of businesses are also adopting a different way of working. If we can have flexible, agile infrastructure in the cloud, we can also speed up the development of environments and applications. Still, also in the cloud, we have to carefully consider service-level objectives and KPIs.

Let’s have a look at the cloud SLA. What are topics that would have to be covered in an **SLA**? The *A* stands for *agreement* and, from a legal perspective, it would be a contract. Therefore, an SLA typically has the format and the contents that belong to a contract. There will be definitions for contract duration, start date, legal entities of the parties entering into the agreement, and service hours. More important are the agreements on the KPIs. What are we expecting from the cloud services that we use and are paying for? And who do we contact if something’s not right or we need support for a certain service? Lastly, what is the exact scope of the service?

These are all quite normal topics in any IT service contract. However, it doesn’t work the same way when we’re using public cloud services. The problem with the SLA is that it’s usually rightsized per business. The business contracts IT services, where the provider tailors the services to the needs of that business. Of course, a lot of IT providers standardize and automate as much as they can to obtain maximum efficiency in their services, but nonetheless, there’s room for tweaking and tuning. In the public cloud, that room is absolutely limited. A company will have a lot of services to choose from to tailor to its requirements, but the individual services are as they are. Typically, cloud providers offer an SLA per service. Negotiating the SLA per service in a multi-cloud environment is virtually impossible.

It’s all part of the service design: the business will have to decide what components they need to cater to for their needs and assess whether these components are fit for purpose. The components—the services themselves—can’t be changed. In IaaS, there will be some freedom, but when we’re purchasing PaaS and SaaS solutions, the services will come out of the box.

The business will need to make sure that an SaaS solution really has the required functionality and delivers at the required service levels.

Common KPIs in IT are availability, durability, **Recovery Time Objective** (**RTO**), and **Recovery Point Objective** (**RPO**), just to name a few important ones. How do these KPIs work out in the major public clouds?

**Defining availability**

Availability is defined as the time when a certain system can actually be used.

Availability should be measured from end to end. What do we mean by this? Well, a VM with an operating system can be up and functioning alright, but if there’s a failure in one software component running on top of that VM and the operating system, the application will be unusable, and hence unavailable to the user. The VM and operating system are up (and available), but the application is down. This means that the whole system is unavailable, and this has some serious consequences. It also means that if we want to obtain an overall availability of 99.9 percent of our application, this means that the platform can’t have an availability below that 99.9 percent. And then nothing should go wrong.

In traditional data centers, we needed to implement specific solutions to guarantee availability. We would have to make sure that the network, compute layers, and storage systems can provide for the required availability. Azure, AWS, and GCP largely take care of that. It’s not that you get an overall availability guarantee on these platforms, but these hyperscalers—Azure, AWS, and Google Cloud—do offer service levels on each component in their offerings.

By way of an example, a single-instance VM in Azure has a guaranteed connectivity of 99.9 percent. Mind the wording here: the connectivity to the VM is guaranteed. Besides, you will have to use premium storage for all disks attached to the VM. You can increase the availability of systems by adding availability, zones, and regions in Azure. Zones are separate data centers in an Azure region. At the time of writing, Azure has 60+ regions worldwide. In summary, you can make sure that a system is always online somewhere around the globe in Azure, but it takes some effort to implement such a solution using load balancing and Traffic Manager over the Azure backbone. That’s all a matter of the following:

* Business requirements (is a system critical and does it need high availability?)
* The derived technical design
* The business case, since the high-availability solution will cost more money than a single-ended VM

**Comparing service levels between providers**

It’s not an easy task to compare the service levels between providers. As with a lot of services, the basic concepts are all more or less the same, but there are differences. Just look at the way Azure and AWS detail their service levels on compute. In Azure, these are VMs. In AWS, the service is called **EC2**—**Elastic Compute Cloud** in full. Both providers work with service credits if the monthly guaranteed uptime of instances is not met and, just to be clear, system infrastructure (the machine itself!) is not available. If uptime drops below 99.99 percent over a month, then the customer receives a service credit of 10 percent over the monthly billing cycle. Google calculates credits if the availability drops below 99.5 percent.

Again, requirements for service levels should come from the business. For each business function and corresponding system, the requirements should be crystal clear. That ultimately drives the architecture and system designs. The cloud platforms offer a wide variety of services to compose that design, making sure that requirements are met. To put it in slightly stronger terms, the hyperscalers offer the possibility to have systems that are ultimately resilient. While in traditional data centers, **disaster recovery** (**DR**) and business continuity meant that a company had to have a second data center as a minimum, cloud platforms offer this as a service.

Azure, AWS, and GCP are globally available platforms, meaning that you can actually have systems available across the globe without having to make enormous investments. The data centers are there, ready to use. Cloud providers have native solutions to execute backups and store these in vaults in different regions, or they offer third-party solutions from their portals so that you can still use your preferred product.

However, it should be stressed once again that the business should define what their critical services and systems are, defining the terms for recovery time and recovery points. The business should define the DR metrics and, even more importantly, the processes when a DR plan needs to be executed. Next, it’s up to IT to fulfill these requirements with technical solutions. Will we be using a warm standby system, fully mirrored from the production systems in a primary region? Are we using a secondary region and what region should that be then? Here, compliancy and public regulations such as the **General Data Protection Regulation** (**GDPR**) or data protection frameworks in other parts of the world also play an important role. Or will we have a selection of systems in a second region?

One option might be to deploy acceptance systems in another region and leverage these to production in case of a failover in DR. That implies that acceptance systems are similar to production systems.

How often do we have to back up the systems? A full backup once a week? Will we be using incremental backups and, if so, how often? What should be the retention time of backup data? What about archiving? It’s all relatively easy to deploy in cloud platforms, but there’s one reason to be extremely careful in implementing every available solution. It’s not about the upfront investment, as with traditional data centers (the real cash out for investment in capital expenditure, or **CAPEX**), but you will be charged for these services (the operational expenditure, or **OPEX**) every month.

In short, the cloud needs a plan. That’s what we will explore in the next sections, eventually in creating a service design.

**Using cloud adoption frameworks to align between cloud providers**

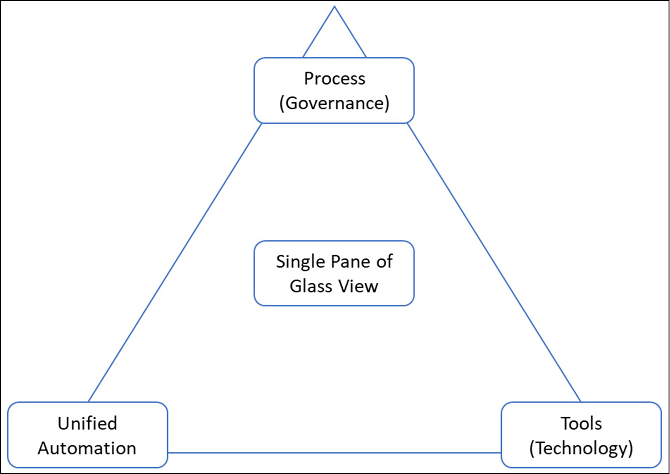
The magic word in multi-cloud is a single pane of glass. What do we mean by that? Imagine that you have a multi-cloud environment that comprises a private cloud running VMware and a public cloud platform in AWS, and you’re also using SaaS solutions from other providers. How would you keep track of everything that happens in all these components? Cloud providers might take care of a lot of things, so you need not worry about, for example, patches and upgrades. In SaaS solutions, the provider really takes care of the full stack, from the physical host all the way up to the operating systems and the software itself. However, there will always be things that you, as a company, will remain responsible for. Think of matters such as IAM and security policies. Who has access to what and when?

This is the new reality of complexity: multi-cloud environments consisting of various solutions and platforms. How can we manage that? Administrators would have to log in to all these different environments. Likely, they will have different monitoring solutions and tools to manage the components. That takes a lot of effort and, first and foremost, a lot of different skills. It surely isn’t very efficient. We want to have a single pane of glass: one ring to rule them all.

Let’s look at the definition of a single pane of glass first. According to TechTarget (<https://searchconvergedinfrastructure.techtarget.com/definition/single-pane-of-glass>), it’s

*a management console that presents data from multiple sources in a unified display. The glass, in this case, is a computer monitor or mobile device screen.*

The problem with that definition is that it’s purely a definition from a technological perspective: just one system that has a view of all the different technology components in our IT landscape. However, a single pane of glass goes way beyond the single monitoring system. It’s also about unified processes and even unified automation. Why is the latter so important? If we don’t have a unified way of automation, we’ll still be faced with a lot of work in automating the deployment and management of resources over different components in that IT landscape. So, a single pane of glass is more an approach that can be visualized in a triangle:



*Figure 4.3: Graphic representation of a single pane of glass*

Do such systems exist that embrace that approach? Yes, there are full IT service management systems such as BMC Helix Multi-Cloud Management and ServiceNow with the Now platform. There are certainly more alternatives, but these are considered to be the market leaders, according to Gartner’s Magical Quadrant for ITSM systems.

ITSM, that’s what we are talking about. The tool—the technology—is one thing, but the processes are just as important. IT service management processes include, as a minimum, the following processes:

* **Incident management**: Tracking and resolving incidents in the platform itself or resources hosted on the platform
* **Problem management**: Tracking and resolving recurring incidents
* **Change management**: Tracking and controlled implementation of changes to the platform or resources hosted on the platform
* **Configuration management**: Tracking the state of the platform and the resources hosted on the platform

The cornerstone of ITSM is knowledge: you need an indisputable insight into what your platform looks like, how it is configured, what types of assets live on your platform, and which assets and resources are deployed on the platform. Assets and resources are often referred to as configuration items and they are all collected and tracked in a **Configuration Management Database** (**CMDB**) or master repository (**Master Data Records**, **MDRs**). Here, the challenge in the cloud really starts. With scalable, flexible resources and even resources that might only live for a very brief period of time in our estate, such as containers or serverless functions, we are faced with the risk that our CMDB or asset repository will never be as accurate as we would like it to be, although the leading ITSM systems have native APIs to the monitoring tools in the different clouds that are truly responsive and collect asset data in (near) real time.

The agility of the cloud makes change management probably the most important process in multi-cloud environments. Pipelines with infrastructure and configuration as code help. If we have a clear process of how the code is forked, tested, validated, and merged back into the master branch, then changes are fully retrievable. If a developer skips one step in the process, we are heading for failure and worse: without knowing what went wrong.

All cloud adoption frameworks stress the importance of governance and processes on top of the technology that clouds provide. All frameworks approach governance from the business risk profiles. That makes sense: if we’re not in agreement about how we do things in IT, at the end of the day, the business is at risk. Basically, the ultimate goal of service management is to reduce business risks that emanate from a lack of IT governance. IT governance and ITSM are a common language among technology providers, for a very good reason.

Back to our *one ring to rule them all*. We have unified processes, defined in ITSM. There are different frameworks for ITSM (ITIL or Cobit, for example), but they all share the same principles. Now, can we have one single dashboard to leverage ITSM, controlling the life cycle of all our assets and resources? We already mentioned BMC Helix and ServiceNow as technology tools. But can we also have our automation through these platforms? Put another way, can we have automation that is fully cross-platform? This is something that Gartner calls **hyperautomation**.

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.

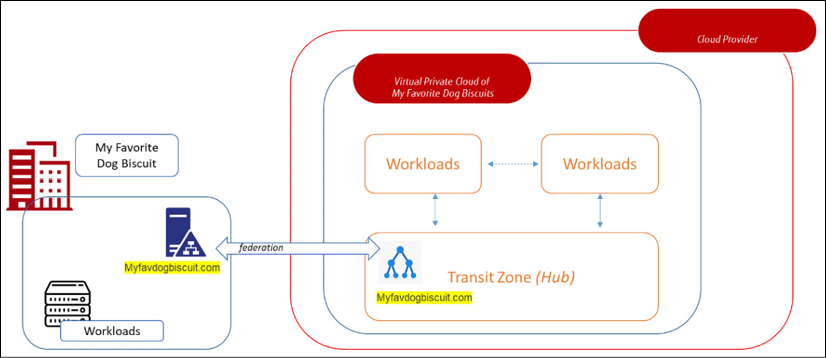
In summary, cloud adoption frameworks from Azure, AWS, and GCP all support the same principles. That’s because they share the common IT service management language. That helps us to align processes across platforms. The one challenge that we have is the single dashboard to control the various platforms 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](http://onmicrosoft.com/) or [aws.amazon.com](http://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.

By way of an example, if a business builds a tiered application with frontends, worker roles, and databases, and defines that as a service to a group of end users, this service needs to be documented separately as a service target.

Next, we will list the relevant requirements that have to be addressed by the service target:

* **Continuity requirements**: This will certainly be relevant for business-critical services. Typically, these are addressed in a separate section that describes RTO/RPO, backup strategies, business continuity plans, and DR measures.
* **Compliance requirements**: You will need to list the compliance frameworks that the company is constrained by. These can be frameworks related to privacy, such as the EU GDPR, but also security standards such as ISO 27001. Keep in mind that Microsoft, AWS, and Google are global companies but are based in the US. In some industry sectors outside the US (this applies typically to EU countries), working with US-based providers is allowed only under strict controls. The same applies to agreements with Chinese providers such as Alibaba, one of the up-and-coming public clouds. Always consult a legal advisor before your company starts deploying services in public clouds or purchasing cloud services.
* **Architectural and interface requirements**: Enterprises will likely have an enterprise architecture, describing how the company produces goods or delivers services. The business architecture is, of course, a very important input for cloud deployment. It will also contain a listing of various interfaces that the business has, for example, with suppliers of components or third-party services. This will include interfaces within the entire production or delivery chain of a company—suppliers, HR, logistics, and financial reporting.
* **Operational requirements**: This section has to cover life cycle policies and maintenance windows. An important requirement that is set by the business is so-called **blackout periods**, wherein all changes to IT environments are halted. That may be the case, for example, at year-end closing or in the case of expected peaks in production cycles. The life cycle includes all policies related to upgrades, updates, patches, and fixes to components in the IT environment.
* **As with the continuity requirements, this is all strongly dependent on the underlying cloud platform**: Cloud providers offer a variety of tools, settings, and policies that can be applied to the infrastructure to prevent the downtime of components. Load balancing, backbone services, and planning components over different stacks, zones (data centers), and even regions are all possible countermeasures to prevent environments from going offline for any reason, be it planned or unplanned. Of course, all these services do cost money, so a business has to define which environments are business-critical so as to set the right level of component protection and related operational requirements.
* **Security and access requirements**: As stated previously, cloud platforms all offer highly sophisticated security tools to protect resources that are deployed on their platforms, yet security policies and related requirements should really be defined by the business using the cloud environments. That all starts with who may access what, when, and why. A suitable RBAC model must be implemented for admin accounts.

Next, we will look at the **Risks, Assumptions, Issues, and Dependencies**(**RAID**) and the service decomposition.

**RAID**

A service design and governance model should contain a RAID log. This RAID log should be maintained so that it always represents the accurate status, providing input to adjust and adapt principles, policies, and the applied business and technical architecture.

**Service decomposition**

The next part is service decomposition, in other words, the product breakdown of the services. What will we be using in our cloud environment?

* **Data components**: What data is stored, where, and in what format, using which cloud technology? Think of SQL and NoSQL databases, data lakes, files, and queues, but also in terms of the respective cloud storage solutions, such as Blob Storage in Azure, S3, Glacier, or Google Cloud Storage.
* **Application components**: Which applications will be supported in the environment, and how are these built and configured? This defines which services we need to onboard and makes sure there’s a clear definition between at least business-critical systems and systems that are not critical. A good method is to have systems categorized, for example, into gold, silver, and bronze, with gold denoting business-critical systems, silver denoting other important production and test systems, and bronze for development systems.
  + However, be careful in categorizing systems. A development system can be critical in terms of finances. Just imagine having a hundred engineers working on a specific system under time pressure to deliver and the development systems become unavailable. This will surely lead to delays and a hundred engineers sitting idle, thereby costing a lot of money. We cannot stress enough how important business cases are.
* **Infrastructure components**: VMs, load balancers, network appliances, firewalls, databases, storage, and so on. Remember that these will all configure items in a CMDB or MDR.
* **Cloud-native components**: Think of PaaS services, containers, and serverless functions. Also, list how cloud components are managed: through portals, CLIs, or code interfaces such as PowerShell or Terraform.
* **Security components**: Security should be intrinsic on all layers. Data needs to be protected, both in transit and at rest, applications need to be protected from unauthorized access, infrastructure should be hardened, and monitoring and alerting should be enabled on all resources.

Typically, backup and restoration are also part of the security components. Backup and restore are elements related to protecting the IT estate, but ultimately protecting the business by preventing the loss of data and systems. Subsequently, for business-critical functions, applications, and systems, a **business continuity and disaster recovery** (**BCDR**) plan should be identified. From the BCDR plan, requirements in terms of RPO/RTO and retention times for backups are derived as input for the architecture and design for these critical systems.

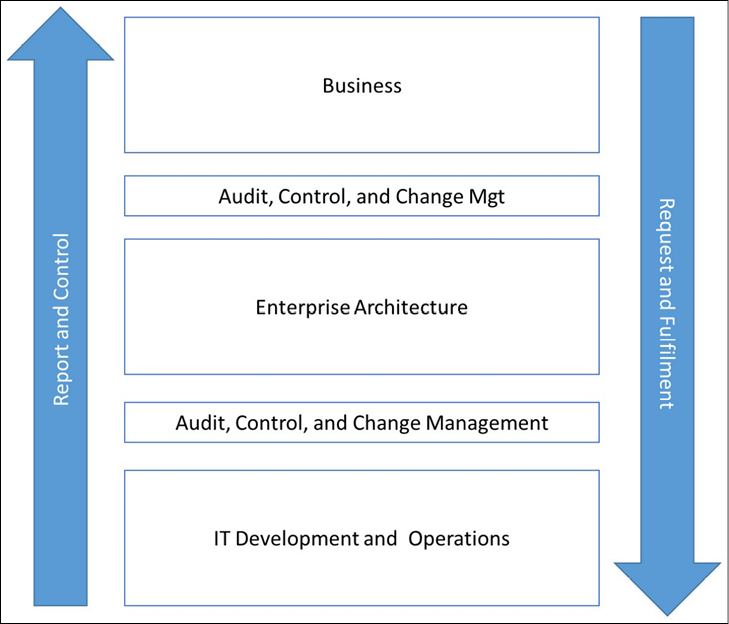
As you can tell, we follow the **The Open Group Architecture Framework** (**TOGAF**) **Architecture Development Method** (**ADM**) cycle all the way: business requirements, data, applications, and the most recent technology. Security is part of all layers. Security policies are described in a separate section.

**Roles and responsibilities**

This section in the service design defines who’s doing what in the different service components, defining roles and what tasks identities can perform when they have specific roles. Two models are detailed in the following sections.

**Governance model**

This is the model that defines the entities that are involved in defining the requirements for the service, designing the service, delivering the service, and controlling the service. It also includes lines of report and escalation. Typically, the first line in the model is the business setting the requirements, the next line is the layer comprising the enterprise architecture, responsible for bridging business and IT, and the third line is the IT delivery organization. All these lines are controlled by means of an audit as input for risk and change control:



*Figure 4.5: Governance model (high-level example)*

Now, this is a very simple, straightforward presentation of a governance model. It can have many variations. For one, IT delivery is often done in Agile teams, working according to DevOps principles. The main principle stays intact, even with Agile teams. Even Agile teams need to be controlled and work under definitions of change management. Just one tip: don’t overcomplicate models; keep it as simple as possible.

**Support model**

The support model describes who’s delivering the support on the various components. Is that solely an internal IT operations division, or do we need to have support contracts from third parties? For that matter, an enterprise will very likely require support from the cloud provider. The support model also defines to what degree that support needs to be provided. In some cases, having the option to have an occasional talk with a subject matter expert from a provider might be sufficient, but when a company has outsourced its cloud operations, full support support is a must.

An important topic that needs to be covered in the support model is the level of expertise that a company requires in order to run its business in cloud platforms. As we already noticed in *Chapter 2*, *Business Acceleration Using a Multi-Cloud Strategy*, there’s a strong tendency to insource IT operations again, since the way IT is perceived has dramatically changed. For almost every company, IT has become a core activity and not just a facilitator. Since more and more companies are deploying services in the cloud, they are also building their own cloud centers of excellence, but with the subject matter expertise and support of cloud providers.

If we follow all the stages so far as a full cycle, then we will end up with a service catalog, describing exactly what services are in scope, how they are delivered, and who’s supporting these services.

**Processes**

This section defines what processes are in place and how these are managed. All processes involved should be listed and described:

* Incident management
* Problem management
* Change management
* Asset management and CMDB (or MDR)
* Configuration management
* Reporting
* Knowledge base

These are the standard processes as used in ITSM. For cloud environments, it’s strongly advised to include the automation process as a separate topic. In the DevOps world that we live in, we should store all code and scripts in a central repository. The knowledge base can be stored in Wikipages in that repository. The DevOps pipeline with the controlled code base and knowledge wikis must be integrated into the ITSM process and the ITSM tooling. Remember: we want to have a single-pane-of-glass view of our entire environment.

There’s one important process that we haven’t touched on yet and that’s the request and request fulfillment. The request starts with the business demand. The next step in the process is to analyze the demand and perform a business risk assessment, where the criticality of the business function is determined and mapped to the required data. Next, the request is mapped to the business policies and the security baseline. This sets the parameters on how IT components should be developed and deployed, taking into account the fact that RBAC is implemented, and resources are deployed according to the services as described in the service decomposition. That process can be defined as request fulfillment.

If different services are needed for specific resources, then that’s a change to the service catalog.

We also need to look at cost management and consistent security postures across our cloud environments. These topics will be extensively discussed in separate chapters.

**Summary**

In this chapter, we’ve explored the main pillars of cloud adoption frameworks, and we learned that the different frameworks overlap quite a bit. We’ve identified the seven stages of cloud adoption up until the point where we can really start migrating and transforming applications to our cloud platforms. In multi-cloud environments, control and management are challenging. It calls for a single-pane-of-glass approach, but, as we have also learned, there are just a few tools—*the one ring to rule them all*—that would cater to this single pane of glass.

One of the most important things to understand is that you first have to look at identities in your environment: who, or what, if we talk about other resources on our platform, is allowed to do what, when, and why? That is key in setting out the governance model. The governance model is the foundation of the service design.

In the last section of this chapter, we’ve looked at the different sections of such a service design. Of course, it all starts with the architecture and that’s what we’ll be studying in the next chapter: creating and managing our architecture.

**Questions**

1. You are planning a migration of a business environment to the public cloud. Would an assessment be a crucial step in designing the target environment in that public cloud?
2. You are planning a cloud adoption program for your business. Would you consider cost management as part of the cloud adoption framework?
3. IAM plays an important role in moving to a cloud platform. What is the environment most commonly used as an identity directory in enterprise environments?

**Further reading**

Alongside the links that we mentioned in this chapter, check out the following books for more information on the topics that we have covered:

* *Mastering Identity and Access Management with Microsoft Azure*, by Jochen Nickel, published by Packt Publishing
* *Enterprise Cloud Security and Governance*, by Zeal Vora, published by Packt Publishing