**3**

**Starting the Multi-Cloud Journey**

HashiCorp published a survey in 2022 that showed that almost 75% of all businesses were executing a multi-cloud strategy. That number is expected to grow to over 86% by 2025. Many more companies claim that multi-cloud is critical for the success of the company. The challenge is how to start a multi-cloud strategy while keeping operations lean and IT infrastructure consistent, but creating development platforms that provide agility to business. It only works with a solid strategy, and that’s what this chapter is about.

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

* Understanding cloud vocabulary
* Planning assessments
* Planning transition and transformation
* Exploring options for transformation
* Executing technology mapping and governance

**Understanding cloud vocabulary**

In IT we have a common language, meaning that the industry has words for common technologies that are used. Whatever technology a company uses, most brands have a commonly accepted name for it. Take networks, for instance. **Wide Area Networks** (**WANs**), **Local Area Networks** (**LANs**), and software-defined LAN: irrespective of the brand that is used to set up a network, the terminology is the same. It’s crucial to have a common understanding of what we mean and what we’re talking about when it comes to the cloud.

That common vocabulary is not a given in the cloud. It already starts with creating a space in a public cloud. In Azure, we’re getting a subscription with a tenant. In AWS, we’re creating accounts, while in Google Cloud, we talk about projects. On the other hand, don’t be distracted by the commercial names that providers give to their services.

Although each provider uses its own terms to refer to similar concepts during implementation, there are certain terms that are often used to talk about the cloud in general. Let’s look at some:

* **CMP**: A **Cloud Management Platform** provides a platform that enables management across various clouds, public and private. It allows management with one single interface, instead of having to go through different cloud portals. It can help in provisioning services, metering and billing, and configuration management of resources that are deployed in various cloud platforms.
* **Cloud Native**: This indicates that applications and software were particularly written for cloud usage.
* **Host**: This is the physical machine that sits in the data center of the cloud provider that hosts the guest environments of customers. Remember the cliché: the cloud is just someone else’s computer. You’re using it.
* **Virtual Machine**: A virtual machine runs on the physical host, but acts as a complete server.
* **Hypervisor**: This allows a host machine to be able to run multiple virtual machines. The technology of hypervisors was essentially the start of the cloud since it allows servers to act as multiple servers. The best-known hypervisor and market leader is VMWare’s vSphere, followed by Microsoft’s Hyper-V, Citrix Xen, and Red Hat Enterprise Virtualization. But there are many more.
* **Hybrid**: Typically, this is to indicate that a cloud platform uses both public clouds, such as AWS or Azure and private clouds that run in the data center of the company itself or a third-party data center.
* **Load Balancing**: If we run multiple resources such as servers, databases, and storage spaces, we need a mechanism that divides the load over these resources. In the cloud, we have many resources, some dedicated to a specific customer, but also a lot of shared resources. Load balancing is essential for spreading traffic and data across the resources.
* **Multi-tenancy**: This is probably the most important term to remember in the cloud, especially in terms of privacy, and keeping our data safe and secure in our own environment, meaning that only authorized users can view the data inside that tenant. Imagine the cloud as a big house with multiple tenants. These tenants will have a room for themselves, but also share facilities. Multi-tenancy indicates that software is running on shared components, where resources are dynamically assigned to software.
* **Resource**: This is the term that is used for components that are used in the cloud to run applications, save data, and execute services. A server is a resource. So is a database or a firewall.
* **Scalability**: Because of virtualization, shared resources, load balancing, and multi-tenancy, the cloud is scalable. Companies use what they need at a certain time. They can scale up when demand is high and scale down in more quiet times. This is the big advantage over environments in privately owned data centers, where equipment (such as servers and storage cabinets) is simply there but sits idle when there’s no demand for it. The inverse is true as well in on-premises: when we need more capacity, such as servers or storage, we will likely need to order the equipment and wait for delivery, whereas in the cloud, we can spin up instances immediately.
* **UI**: The user interface. At the end of the day, operators and developers must be able to work with the resources in the cloud. This is done through the UI in most cases, a portal with all the services that can be deployed in a cloud platform. Having said that, most developers and operators use other mechanisms to manage workloads in the cloud, since a portal with a graphical UI is too slow for large-scale operations. Then scripting comes into play, for instance, using command-line interfaces with PowerShell that can be used in Azure and AWS, and GCP commands in GCP. To summarize, public clouds provide a UI for users and APIs for large-scale operations. In the latter case, we will need to script in, for instance, Powershell to interact with the APIs.

Speaking the same language helps in understanding what specific clouds we are talking about and how we can use them in the most optimal way. It’s like going on a real journey: it helps when we can speak at least some basic lines in the language of the country we are traveling to so that we are sure that we reach our planned destination. We have to plan our journey. Hence, the next step is planning and executing an assessment to validate if we are ready to migrate to the cloud.

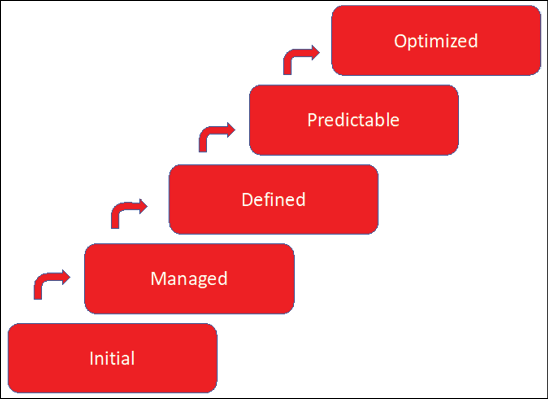
**Planning assessments**

You have to know where you’re coming from in order to define where you’re going. There’s absolutely no use in building a cloud landing zone without knowing what will land on the platform. The next question is how applications should be migrated to the platform—if the business case is positive and the decision has been made to start the migration. Note that migration is one of the cloud adoption paths. We can also develop cloud-native solutions, but in that case, companies will likely have to redesign and rebuild applications. Migration is typically about moving existing applications to the cloud, typically referred to as rehosting or replatforming. Part of the migration can be to start using PaaS services, for instance, managed database services from the cloud. However, the application itself usually remains as it is. We will talk about this in more detail later in this chapter.

In this section, we will assess the readiness of the organization to adopt the cloud, just like the Cloud Adoption Framework advises. Relevant questions include “Who are all the stakeholders that need to be involved in the activities?” Most importantly, “Will the migration impact the customer, thus the business, and in what way? What are the drivers to start the transition and transformation to a cloud platform?” These are just a few questions that must be answered before we start planning the actual activities. They will show if the organization is ready to start adopting cloud technology. The assessment must reveal the business advantages, the effort the business needs to invest, and if the right people with the right skills are available.

Is an assessment necessary? The short answer is: yes—without exceptions. Assessments will also reveal the level of maturity of an organization. This level of maturity helps an organization to understand whether they are ready to move to the cloud at all.

Perhaps this is a good spot to briefly talk about maturity. For starters, a mature IT organization doesn’t necessarily mean a mature organization working in the cloud. Most maturity models work according to five levels. *Figure 3.1* shows an example, in this case, the **Capability Maturity Model**, or **CMM**.



*Figure 3.1: The Capability Maturity Model (CMM)*

Level 1—initial is the lowest level, where processes are poorly controlled and outcomes are highly unpredictable. In the cloud, this would mean that organizations would simply start building environments in a cloud without proper processes or defining what the result should be. Level 5 is the highest level, where organizations can focus on improvements since projects and management are well-defined, processes are under control, and outcomes are predictable and measurable.

Most organizations are somewhere between levels 3 and 4. But are these organizations not working on improvements? They are, but not as part of the standing organization. Typically, improvements and innovations are done separately from production and need handover procedures to implement into production.

One more time: enterprises that have well-defined and controlled IT organizations aren’t necessarily also well-equipped for starting a transition into the cloud. To put it in different words: the cloud is a different beast. Where IT organizations run the entire platform, including the hosting infrastructure, they are now running on “someone else’s computer.” Here’s where the segregation of duties plays a very important role. Cloud providers are responsible for the cloud (the data center), and customers are responsible for what’s in the cloud (the virtual servers in the data center and the data). Note that this is the case for IaaS. For PaaS and SaaS, this changes. For example, in the case of SaaS, the provider is responsible for the entire solution.

Is an organization ready to start working in such a model? Notice that this is not a technical question: we’re not starting with mapping the existing technology to the cloud. We’re starting with organizational readiness since the operating model in the cloud is fundamentally different from privately owned data centers. Readiness is the first topic in any assessment.

Let’s explore the common steps in an assessment:

1. **Set the scope**. This should not be executed from the technology but from the business perspective. What are the business functions that would benefit from a transformation and the usage of cloud technology?
2. **Identify all infrastructure components**. This results in a complete list of all infrastructure, such as network equipment, servers, storage, and security appliances. Can we map these components to solutions in the cloud? What solutions do we need to implement infrastructure in the cloud?
3. **Identify all applications and application components**. This results in an application mapping showing what business functionality is fulfilled by what application and specific application components. Are applications cloud-ready, can they be migrated to cloud infrastructure, or do we need to refactor, replatform, or even redesign applications before we can host them in the cloud? What would be the effort to get applications cloud ready?
4. **Identify security requirements, including existing policies and security guardrails**. What security policies are in place and how do these translate in the cloud? Are existing policies including compliance statements, and if so, do cloud solutions adhere to these compliance rules?
5. **Identify all stakeholders that must be involved in migrating to the cloud, including business and application owners**.

The information that is discovered through these steps could determine if an organization is ready to move to the cloud or if they need more time. This will drive the business case.

One of the challenges that many enterprises face is the availability of resources. It’s recommended to assess the availability of resources: does the company have the right people with the required skills to execute the transformation? Competition is fierce in the war on talent; this must be addressed in the earliest stage. This will also impact the business case; timelines and budgets will be heavily put under pressure if resource planning is not considered.

**Executing technology mapping and governance**

Companies have an ambition and a strategy to fulfill that ambition. In the modern, digital company, technology will undoubtedly play an important role in fulfilling that ambition. But, as with every other aspect within the governance of a company, the deployment and management of technology needs proper planning. First, technology must be assessed against the goals a company wants to achieve. In other words: technology must add value to the business goals. Second, a company must be ready to adopt new technology. This means that it must have trained, skilled staff that is able to work with it. With that, we have identified the two main blockers to the successful implementation of technology, including the cloud:

* It might not add value to the business
* There are no human resources available who can work with it

Technology mapping can help here. It typically starts with defining the use cases: what will the technology be used for? Next, it’s assessed against the existing portfolio to determine how the technology must be implemented and integrated within the processes and technology base of the company. The final step is actual planning, wherein the adoption of change and training are key elements.

It starts with the ambition and the strategy. In a world where markets and customer demands change continuously, business agility is a must. Companies must be able to respond to changes fast and act accordingly. Innovations can’t be neglected, but it’s important to pick the right innovations.

**Keeping track of innovation**

Cloud technology evolves fast. Remember that AWS started in 2006 with its **Simple Queue Service** (**SQS**), a messaging service that could be publicly used. Today, AWS offers well over two hundred services from data centers all over the world. New services are launched every year. How do companies keep track of all the innovations?

First of all, not every new service might be beneficial for your company. Services need to fit the business ambition, strategy, and goals—as with everything. Having said that, most innovations can be categorized as features that either make working in the cloud easier (automation), more efficient (cost and efficient use of multiple data sources), or open up new possibilities for businesses. An example of the latter is services that enable the use of artificial intelligence or even quantum simulation.

It’s important to have a clear vision of where a business is going. In the digital era, that is already very hard. Customer demand changes constantly and at an ever-increasing speed. Innovations can surely help businesses to fulfill these demands. The challenge is how to assess and judge if innovations are really need-to-haves or nice-to-haves.

The major cloud providers organize yearly conferences where they announce major new features and releases of new technology. There’s something to be aware of with these events: they are targeting tech-savvy audiences.

**Microsoft Ignite**: This is the event for technology professionals and developers working with Microsoft technologies, including Azure. It focuses on new technology that Microsoft has developed and released in Azure.

**Microsoft Inspire**: This is the business event for partners offering services using Microsoft technologies, such as Azure. The conference focuses on how to use Azure services to grow and manage the business and is less technical than Ignite.

**AWS Re.invent**: The technology conference of AWS, which annually shows new products and releases.

**AWS business events**: This platform is for business leaders, providing opportunities to discuss how AWS can help expand the business.

**Google I/O**: The big technology conference of Google covers more than just Google Cloud. It addresses every technology that Google provides, including Android and Google services.

**Google Cloud Next**: The main event for Google Cloud.

**Oracle Cloud World**: The main event to learn everything about new releases and features in Oracle Cloud Infrastructure.

These are the conferences where the latest information about products and platforms is shared. Next to this, there are user communities around the world that are worthwhile attending. Experts Live, focusing on Microsoft technologies, is a good example of such a user event. Lastly, social media such as Twitter, Mastodon, LinkedIn, and Reddit are great sources to keep track of innovations.

**Adopting innovations**

Adopting innovations is a matter of culture in the first place, but the most important thing is that innovations must be triggered by the business. Innovations must lead to added value. The challenge for every innovation is to question what benefit it will bring to the customers of a company, and what added value is involved. This can be a direct benefit leading to a better customer experience, but also as a result of optimized processes within the company. In the end, that will also lead to improved customer experience.

Let’s have a look at the common ambitions of modern companies. These are commonly grouped into customer experience, sustainability, and financial efficiency. If these are the greater ambitions of a company, then innovations must be challenged against these ambitions.

Is it improving the customer experience—will it make customers happier?

Is it improving the sustainability of the enterprise and/or society at large?

Given the overall business ambition, is it worth the investment, and how will it impact financial performance?

Let’s use an example. Serverless functions were a great innovation in cloud technology. When used in event-driven use cases, it’s likely that they improve the customer experience since actions are immediately started at a customer trigger and delivery of services commences without manual interventions. They also improve sustainability since there’s no use for heavy VMs that need to be deployed and managed. Fewer resources are invoked, resulting in less impact. Are they worth the investment? The answer is probably yes, given the fact that they will add value to the business.

But, it’s also likely a huge change in the architecture of systems. That must be taken into account: it will require skilled resources and effort to adopt the innovation and make the required changes in the landscape. That is part of the total investment. Then we must agree on the importance of customer experience and reaching sustainability goals.

One important aspect that we must consider in introducing innovations is that not everyone will be willing to adopt these. Operators have a task to keep systems as stable as possible. Innovations causing changes to systems might lead to failures and even outages of systems, resulting in a lot of extra work for them. This is the reason to always bring developers and operators together in introducing and implementing innovations, indeed in a DevOps manner. They all need to be on the same page.

That is culture: reaching common agreements on the business goals and a common understanding of the impact of change by adopting innovations. There will be a lot more on this topic during the course of this book.

To summarize, innovations must be significant to the business. They must lead to added value and add to the overall business ambition of a company. With that, we are at defining that ambition in a North Star and how multi-cloud can fit into that.

**Defining roadmaps and business alignment**

There’s a good chance that you have heard about the North Star architecture. It’s commonly used to define the future ambition of an enterprise, especially in digital transformation. Almost every enterprise has defined a North Star. It sets the ambition and lets teams iterate toward the realization of that ambition: it’s the fundamental idea of agile working. The architecture evolves in every iterative step, adapted, and refactored whenever necessary to move toward the North Star.

However, teams need guidance and guardrails. The risk of iterative working and only having an ambition defined in the North Star is that teams develop code and applications without guidance on interoperability and integration throughout systems. Or, teams spend a lot of time developing something that another team has already done. They might develop a feature that is readily available as a service in a cloud platform.

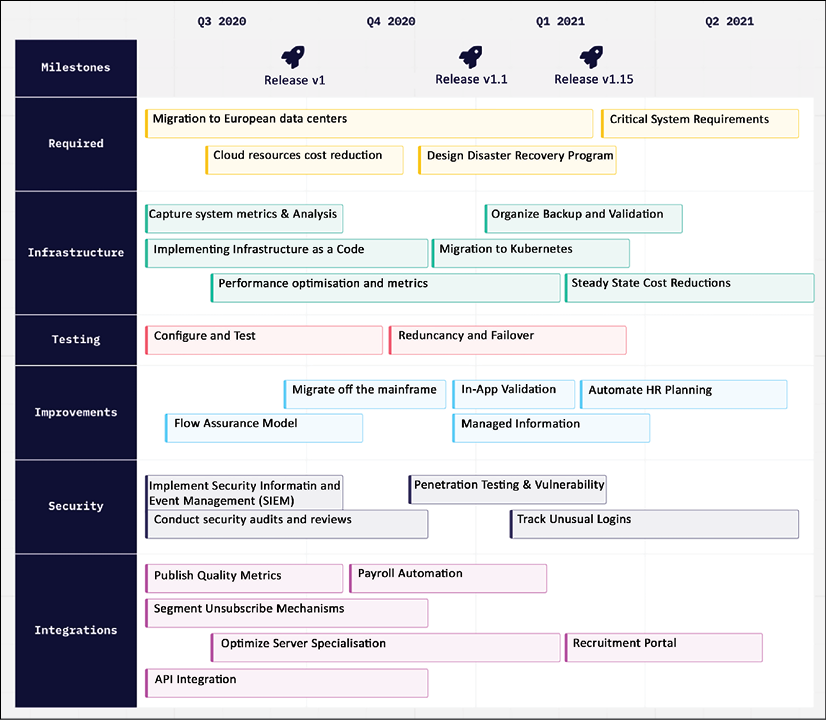
Roadmaps will provide guidance, but only if the roadmap itself is “business integrated.” What do we mean by that? Business ambition and goals must be aligned with the capabilities of the enterprise and next, the technology stack of the enterprise. Combined, these should converge into an actionable strategy telling teams what to use to build specific business functionality.

So, in building roadmaps, we start with the business model and the business functionality. The business model provides insights into how the enterprise works, translated into business functionality. This maps to application functionality and the usage of data inside the functionality. What business function does an application serve and what data does the function need, when, and why? Lastly, the technology mapping is done.

What sort of databases would serve data processing best? Which technology does the enterprise need to deliver application, and thus business, functionality? In a roadmap, this can then be matched with announced propositions from cloud providers and other suppliers. But, it must be clear from the roadmap how a new feature, release, product, or service will enhance the functionality.

In short, a technology roadmap defines strategic planning to achieve specific business goals and ambitions using technology. Technology roadmaps include not only the technology itself but map these to business functions.

Architects are encouraged to use templates to build roadmaps. An example is provided in the following diagram. Tools such as Miro are very useful in building roadmaps.



*Figure 3.2: Example of a Miro board showing a technology roadmap (*[*https://miro.com/templates/technology-roadmap/*](https://miro.com/templates/technology-roadmap/)*)*

Roadmaps will help architects to define strategies for migrating or updating software systems, and planning the development of new technology or other projects. But architects have to keep in mind that there’s one starting point, and that’s the business. In the next chapter, we will discuss service designs in the cloud and how business **key performance indicators** (**KPIs**) and **service-level agreements** (**SLAs**) will help companies get the best out of the cloud. Cloud providers offer excellent tools for that with the **Cloud Adoption Framework** (**CAF**) and **Well-Architected Framework** (**WAF**). We will learn how to use these in architecture.

**Planning transition and transformation**

It must be clear that the planning of the transition and transformation to the cloud starts with the business case. Architects are advised to follow the business-data-application-technology rule to design the roadmap for applications that are designated to be moved to cloud platforms. But before companies can really start migrations, there’s a lot of work that must be done.

Assuming that, from the assessment, we have a clear overview of the **current mode of operations**, the **CMO**. From the business case, the company also has set the ambition, goals, and strategy to transform the business to the **future mode of operations** (**FMO**)—and the rationale as to why it’s necessary or desirable to transform. As said, this goes beyond technology. The transition and transformation plans should include at least the following topics on the various tiers as shown in the diagram:



*Figure 3.3: Simplified enterprise governance model*

The highest tier is strategic. This is the tier where the actual business strategy is defined, the topic of the previous chapter. This is also the level where all the governance processes are defined to help govern and control all activities in migration and, eventually, operations.

The next tier is tactical. An enterprise that is moving applications—or starting to develop applications—in public cloud platforms will have to think about service levels and KPIs. There will be contracts and agreements with the cloud providers that must match the business objectives. All major cloud providers offer SLAs. These contracts define things such as uptime guarantees and credit policies. These are typically legal documents; hence it’s advised to involve a legal counselor to study these documents.

The SLAs for Microsoft Azure can be retrieved from <https://www.microsoft.com/licensing/docs/view/Service-Level-Agreements-SLA-for-Online-Services?lang=1>; for AWS, refer to <https://aws.amazon.com/legal/service-level-agreements/>, and for GCP, to <https://cloud.google.com/pubsub/sla>.

The final tier is the operational tier. This is the tier where all operational processes are defined: build, test, deploy, operate, and monitor. In every operational process, it must be clear how cloud services can and should be used. This includes the management of infrastructure, data, and applications.

This means that in the planning of transition and transformation, the architect must start with translating the business ambition and goals to tactical parameters, and lastly, to technical solutions.

A transition plan will consist of the following items to cover it all:

* **Discovery of assets**: We need to have a complete and comprehensive overview of the environment and how it’s serving the business processes. The latter is important: there will be easy, standalone applications that can be migrated to the cloud without a severe impact on critical business processes. There will be, without a doubt, applications and data that are crucial to the business. When the current environments suffer from outages, the lack of access to these crucial applications and data can cause a lot of damage, even when outages are planned. From the discovery, it must be clear from the business functionality and criticality what environments are crucial. This will influence the migration strategy.
* One other important topic in discovery is that we need to establish all interactions in the environment. What workload is connected to other workloads? This involves routing, database usage, (shared) storage, backups, and the security settings in an environment.
* Last but not least, we need to identify how the assets are operated and by whom.
* **Assessing the assets against the business plan**: The architect needs to define a target architecture and a target operating model. Both must comply with some principles before the actual build can start. A multi-cloud strategy and architecture must ensure consistency across the targeted platforms. This is not only a matter of avoiding the lock-in in a specific cloud but more so a matter of reducing complexity in managing workloads. We will be distributing workloads across platforms, even in various service models such as IaaS, PaaS, and SaaS. Then, it’s important to take quality attributes into account:
  + Reliability
  + Maintainability
  + Usability
  + Portability
  + Correctness
  + Efficiency
  + Integrity or security
  + Testability
  + Flexibility
  + Reusability
* If the architect works against these quality attributes, the company will get the benefits of multi-cloud:
  + **Best of breed solutions**: Since this is reason number one to start a multi-cloud journey. We want to use the best of the various worlds and have the best solution for our enterprise proposition, delivering value to the customers.
  + **Improved resiliency**: Workloads are distributed across various platforms. This will decrease the risk of failures and outages with a severe impact on the business.
  + **Business agility**: Multi-cloud leverages agility, by taking advantage of the opportunity to choose the right tools and solutions for business demand.
* There are two more things that architects already must consider in this stage:
  + **Security**: Here’s the statement: already, public clouds are extremely well secured. Cloud providers don’t have a choice in securing their platforms, with thousands of customers that work with their services and run (critical) business processes. But the provider is responsible for the cloud, and the customer for what’s in the cloud. AWS, Azure, and GCP offer a wide variety of security tools, but it’s up to the customer to use them wisely.
  + **Cost control**: Companies must be aware of the costs that they are generating by using cloud services. The cloud can be complex in terms of costs, metering, and billing. Services constantly change, and new features are continuously released, making it hard to always have a proper understanding of where costs are coming from. This is the domain of FinOps, which we will discuss in depth in this book.
* **Planning**: This will involve all activities needed to actually perform the migration and the transformation of the workloads. Typically, the migration is planned in waves, where Wave 0 is used to build and test the landing zone and the migration procedures, as defined in the target architecture and target operating model. It’s advised to have simple applications in Wave 1 and more complex environments in later waves.

We now have a plan and we can start the build. It’s the topic of the next section.

**Starting the build**

One of the key decisions a company makes when defining cloud strategy is which cloud providers to choose. Then, they should build landing zones in each cloud of choice. This should include the basics, such as communications, policies, security, monitoring, business continuity and disaster recovery, processes, and training. From there, the business can start working on planning and building specialized environments per workload. Remember: there’s no point in building and configuring a landing zone without knowing what will be landing on it.

What is a landing zone? Typically, a cloud landing zone is defined as the pre-configured basic environment that is provisioned in a public cloud so that it’s ready to start hosting applications. Landing zone is a common term, but—remember the first section in this chapter about the common language—all cloud providers have different names and services to create landing zones.

There’s an important step before we start configuring the landing zone itself: making sure that we can reach the landing zone. We need connectivity.

**Setting up simple connectivity**

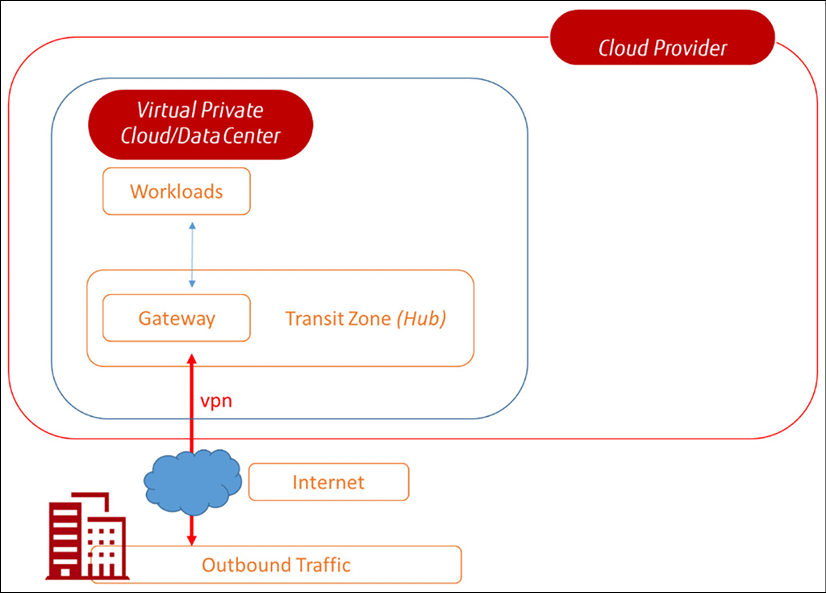
You can make a subscription to any cloud provider and start working in it, but a company would want to have an enrolment to which its workers can securely connect from the company’s domain to a specific cloud service. Basically, there are three options to enable that connection: a **virtual private network**(**VPN**), direct connections, and using a fully managed broker service from a telecom company or connectivity partner. In the next sections, we are going to have an in-depth look at each of these options.

One of the most used technologies is the VPN. In essence, a VPN is a tunnel using the internet as a carrier. It connects from a certain IP address or IP range to the IP address of a gateway server in the public cloud.

Before we get into this, you have to be aware of what a public cloud is. If you as a business deploy services in Azure, AWS, **GCP**, or any other public cloud (remember, there are more public clouds, such as Oracle, OpenStack, IBM Cloud, and Alibaba, and the basic concepts are all more or less the same), you are extending your data center to that cloud. It, therefore, needs a connection between your data center and that extension in the public cloud.

The easiest and probably also the most cost-efficient and secure way to get that connection fast is through a VPN. The internet is already there, and all you would have to do in theory is assign IP addresses or the IP range that is allowed to communicate to that extension, creating a tunnel. That tunnel can be between an office location (site) or from just one user connecting to the cloud. The latter is something we refer to as a point-to-site or site-to-site VPN.

In the public cloud itself, that connection needs to terminate somewhere, unless you want all resources to be opened up for connectivity from the outside. That is rarely the case, and it’s certainly not advised. Typically, a business would want to protect workloads from direct and uncontrolled external connections. When we’re setting up VPNs, we need to configure a zone in the public cloud with a gateway where the VPN terminates. From the gateway, the traffic can be routed to other resources in the cloud, using routing rules and tables in the cloud. It works the same way as in a traditional data center, where we would have a specific connectivity zone or even a **demilitarized zone** (**DMZ**) before users actually get to the systems. The following architecture shows the basic principle of a VPN connection to a public cloud:



*Figure 3.4: The basic architecture of VPN connectivity*

One remark: there are other ways to connect to public clouds, including direct, dedicated connectivity with, for instance, Azure ExpressRoute, AWS Direct, and Google Direct Connect. These are typically considered by enterprises in case of large, mission-critical deployments in public clouds and substantial transfer of data that requires high performance. This type of connection is also recommended—sometimes even required—to adhere to compliance statements or to ensure reliability and consistent latency.

The next step is to launch the landing zone, our workspace in the cloud.

**Setting up landing zones**

In Azure, the landing zone starts with a subscription. AWS deploys accounts and GCP provisions projects. But they all do sort of the same thing, and that is configuring a basic environment where you can start working with the respective services that these providers offer. Looking at frameworks such as Well-Architected and Cloud Adoption, these landing zones offer:

* Security controls
* **Identity and Access Management** (**IAM**)
* Tenancy (“workspace” allocated to a specific customer)
* Network configurations
* Resources related to operations such as monitoring and backup

With landing zones, customers get a standardized environment. Yet, it’s good practice to think about setting up landing zones so that they can be managed from day one. The phases for setting up landing zones are:

* **Design**: Defining the basic parameters for security controls, IAM, and networking.
* **Deploy**: In Azure, we can use Azure Blueprints to deploy a landing zone. It offers standardized templates to roll out a landing zone, with the possibility to integrate design principles. In AWS, the service is simply called AWS Landing Zone, which sets up all the basic services in AWS using Cloud Formation templates and services such as CloudTrail for monitoring. Google offers the Google Deployment Manager, which allows developers to deploy templates in GCP using YAML, Python, or Jinja2.
* **Operate**: As soon as a customer starts building things, the landing zone will expand. It needs to be managed, just as any other infrastructure. If companies run multiple environments in the mentioned clouds, it’s wise to set up tools that make operations across these multiple environments easier. We will discuss this in the section about keeping infrastructure consistent.
* **Update**: Updates from operations must be looped back into the design so that architectures are continuously improved and stay consistent, documented, and transferable.

Following the standardized approaches of the cloud providers, the landing zone will enable organizations to consistently manage:

* Resources in the cloud platform
* Access
* Monitoring
* Costs
* Governance and security

This chapter focused more on the generic principles for setting up landing zones. In *Chapter 6*,*Controlling the Foundation Using Well-Architected Frameworks*, we will discuss the various landing zone propositions in AWS, Azure, GCP, Alibaba, and OCI in more detail.

With a landing zone in place in our preferred cloud or multiple clouds, we can start thinking of migrating or deploying workloads.

**Exploring options for transformation**

In developing and migrating workloads to the cloud, there are a number of options that architects must consider from the beginning. In this section, we will elaborate on these choices.

**From monolith to microservices**

A lot of companies will have technical debt, including monolithic applications. These are applications where services are tightly coupled and deployed as one environment. It’s extremely hard to update or upgrade these applications; updating a service means that the whole application must be updated. Monolithic applications are not very scalable and agile. Microservices might be a solution, wherein services are loosely coupled.

Transforming a monolithic application to microservices is a very cumbersome process. First of all, the question that must be answered is: is it worthwhile? Does the effort and thus costs weigh up to the benefits of transformation? It might be better to leave the application as-is, maybe lift-and-shift it to the cloud, and parallel design, build, and deploy a replacement application using a microservices architecture.

With that, we have identified the possible cloud transformation strategies that we discussed in *Chapter 2*,*Collecting Business Requirements:*

* Rehost
* Replatform
* Repurchase
* Refactor
* Rearchitect
* Rebuild
* Retire
* Retain

Microservices typically involve a rearchitect and refactor. The functionality of the application and the underlying workloads is adapted to run in microservices. That includes the rearchitecture of the original monolithic application and refactoring the code. This might sometimes be a better option, with the development of cloud-native applications replacing the old application, especially when the original application is likely to consume a lot of heavy—costly—resources in the cloud or prevent future updates and upgrades.

Enterprises can make the decision to retain the old environment for backup or disaster recovery reasons. This will definitely lead to extra costs: costs of having to manage the old environment and investing in the development of a new environment. Part of the strategy can also be “make or buy” with either in-house development or buying software “off the shelf.” It all needs to be considered in drawing the plans.

More technology has been emerging to create cloud-native architectures, moving applications away from the classical VMs to containers and serverless. We’ll review this in the next sections.

**From machines to serverless**

Public clouds started as a copy of existing data centers, with servers, storage, and network connectivity like companies had in their own data centers. The only difference was that this data center was operated by a different company that “rented” equipment to customers. To enable a lot of customers to make use of the data center, most of the equipment was virtualized by implementing software that divided a server into multiple, software-defined servers. One server hosted multiple customers in a multi-tenant mode. The same principle was applied to storage and network equipment, enabling very efficient usage of all available resources.

Virtual machines were the original model that was used by customers in the public cloud. A service was hosted on a machine with a fixed number of CPUs, memory, and attached disks. The issue was that, in some cases, services didn’t need the full machine all of the time, but nonetheless, the whole machine was charged to the customer for that time. There were even services that only were short-lived: rapidly scaled up and down again, as soon as the service was not running anymore. Especially in microservices, this is becoming common practice. Event-based architectures are where a service is triggered by an action of a customer and stopped as soon as the action has been executed. Full-blown virtual machines are too heavy for these services.

Serverless options in the public cloud are a solution for this. In the traditional sense, you use your own server on which your own software runs. With a serverless architecture, the software is run in the cloud only when necessary. There’s no need to reserve servers, saving costs. Don’t be fooled by the term “serverless” as there are still servers involved, but this time, the service only uses a particular part of the server and only for a short amount of time. Serverless is a good solution when an organization doesn’t want to bother about managing the infrastructure.

But the biggest advantage of using serverless options is the fact that it can help organizations in becoming event-driven. With microservices, there are software components that focus on one specific task, such as payments or orders. These are transactions that follow a process. Serverless functions each perform their own step in the process, only consuming the resources the function needs to execute that specific task in the process. Then, serverless is a good option to include in the architecture.

Major public clouds have these solutions: Azure Functions, AWS Lambda, and Google Cloud Functions.

**Containers and multi-cloud container orchestration**

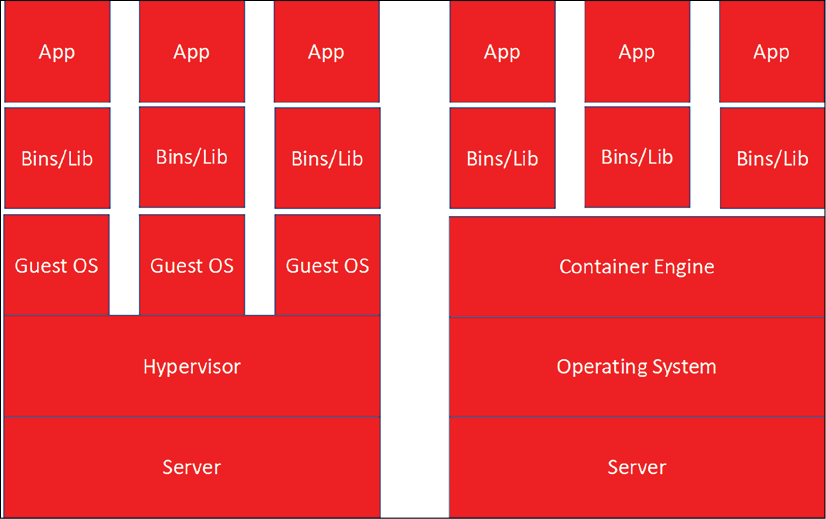
Serverless is a great solution to run specific functions in the cloud, but they are not suitable to host applications or application components. Still, companies want to get rid of heavy virtual machines. VMs are generally an expensive solution in the cloud. The machine has a fixed configuration and runs a guest operating system for itself. So, the hosting machine is virtualized, allowing multiple workloads to run on one machine. But these workloads that run in a VMs, still require their own operating system. Each VM runs its own binaries, libraries, and applications, causing the VM to become quite big.

Containers are a way to use infrastructure in a more efficient way to host workloads. Containers work with a container engine and only that engine requires the operating system. Containers share the host operating system kernel but also the binaries and libraries. This makes the containers themselves quite light and much faster than VMs. When an architecture is built around microservices, containers are a good solution.

Containers are a natural solution to run microservices, but there are other scenarios for containers. A lot of applications can be migrated to containers easily and with that, moved to the cloud quickly—making containers a good migration tactic.

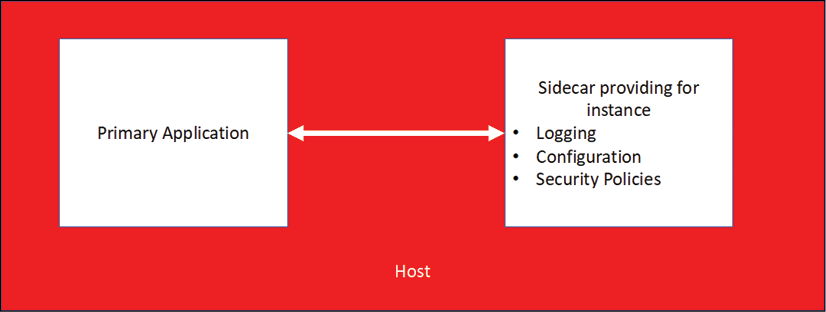
Each container might run a specific service or application component. In case of upgrades or updates, only a few containers might be “impacted.”

The following diagram explains the difference between a VM and a container:



*Figure 3.5: Virtual machine (left) versus containers (right)*

This is a good point to explain something else in working with a container architecture: sidecars. Sidecar containers run along with the main container holding a specific functionality of the application. If we only want to change that functionality but nothing else, we can use sidecar containers. The sidecar container holds the functionality that shouldn’t be changed, while the functionality in the main container is updated. The following diagram shows a simple example of a sidecar architecture:



*Figure 3.6: Simple architecture for a sidecar*

There’s a catch to using containers, and that’s the aforementioned container engine. You need a platform that is able to run the containers. The default industry standard has become Kubernetes.

With Kubernetes, containers are operated on compute clusters with a management layer that enables the sharing of resources and the scheduling of tasks to workloads that reside within the containers. Resources are compute clusters, a group of servers –commonly referred to as nodes—that host the containers. The management or orchestration layer makes sure that these nodes work as one unit to run containers and execute processes—the tasks—that are built inside the containers.

The cluster management tracks the usage of resources in the cluster such as memory, processing power, and storage, and then assigns containers to these resources so that the cluster nodes are utilized in an optimized way and applications run well.

In other words, scaling containers is not so much about the containers themselves but more about scaling the underlying infrastructure. Kubernetes uses Pods, enabling the sharing of data and application code among different containers, acting as one environment. Pods work with the share fate principle, meaning that if one container dies in the Pod, all containers go with it.

All major cloud providers offer solutions to run containers using Kubernetes:

* **Azure Kubernetes Services** (**AKS**): The managed service to deploy and run Kubernetes clusters in Azure. Azure also offers Azure Container Apps and Azure Container Instances as serverless options.
* **Elastic Kubernetes Services** (**EKS**): The AWS-managed service for Kubernetes platforms in AWS. EKS Anywhere uses EKS Distro, the open-source distribution of Kubernetes. Since it’s a managed service, AWS takes care of testing and tracking Kubernetes updates, dependencies, and patches. To be clear, the same applies to AKS. AWS also offers a serverless solution to run container environments: Fargate. This removes the need to provision and manage servers, and simply allocates the right amount of compute, eliminating the need to choose instances and scale cluster capacity.
* **Google Kubernetes Engine** (**GKE**): The managed service of Google Cloud to deploy and run Kubernetes clusters in GCP.
* **Alibaba Cloud Container Service for Kubernetes** (**ACK**): The managed service of Alibaba Cloud to deploy and run Kubernetes clusters on Alibaba Cloud.
* **Oracle Container Engine for Kubernetes** (**OKE**): The managed container service that we can use in OCI. This service also includes serverless options with OKE Virtual Nodes.

All the mentioned providers also offer unmanaged services to run containers and container orchestrations, but the advantage of managed services is that the functionality of, for instance, scaling and load balancing across clusters is completely automated and taken care of by the provider.

The Kubernetes clusters with Pods and nodes must be configured on that specific platform, using one of the services mentioned above. There are technologies that provide tools to manage Kubernetes clusters across various cloud platforms, such as VMWare Tanzu, NetApp Astra, and Azure Arc:

* **VMWare Tanzu**: This is the suite of products that VMware launched to manage Kubernetes workloads and containers across various cloud platforms. The offering was launched with Tanzu Mission Control but has evolved over the past years with offerings that allow for application transformation (Tanzu Application Platform) and cloud-native developments (Tanzu Labs).
* **NetApp Astra**: NetApp started as a company that specialized in storage solutions, specifically in **network attached storage** (**NAS**), but over the years, NetApp evolved to a cloud management company with a suite of products, including Astra, that allows management of various Kubernetes environments.
* **Azure Arc**: Azure Arc-enabled Kubernetes allows you to attach and configure Kubernetes clusters running anywhere. You can connect your clusters running on other public cloud providers (such as GCP or AWS) or clusters running on your on-premises data center (such as VMware vSphere or Azure Stack HCI) to Azure Arc.

The interesting part of these products is that they are also able to manage lightweight Kubernetes, known as K3S, in environments that are hosted in on-premises private stacks, allowing for seamless integration of Kubernetes and container management through one console.

**Keeping the infrastructure consistent**

Microservices, serverless, containers, and legacy environments that run virtual machines in a more traditional way all need to be operated from the cloud. The big challenge is to keep the infrastructure consistent. In this section, we will briefly discuss methodologies and tools to achieve this.

The preferable way of keeping infrastructure consistent is by working through templates. Such a template contains all the configurations with which an infrastructure component should comply. We can take a virtual machine as an example. A VM can be deployed straight from the marketplace of a cloud provider. Typically, companies have specific demands for servers: they must be configured with specific settings that define the configuration of the operating system, level of access, and security parameters of the server. First, we don’t want to do this manually every time we enroll a server, and second, if we do it manually, the chances are that there will be deviations. Hence, we use templates to automate the enrollment of the desired state of servers and to keep all servers consistent with policies.

Let’s look at an example of how a template for a VM could look. Be aware that this list is not meant to be exhaustive:

* Sizing of the VM
* Operating system
* Configuration parameters of the operating system
* Access policies
* Network settings
* Workgroup or domain settings
* Security parameters
* Boot sequence

This can, of course, be done for every component in our infrastructure: storage, databases, routers, gateways, and firewalls, for example. There are a couple of methods to create templates. The two common ones are:

* Manual configuration and saving the template in a repository
* Cloning a template from an existing resource

There are a number of tools that can help in maintaining templates and keeping the infrastructure consistent:

* **Terraform**: This is an open-source tool by HashiCorp and became the industry standard for **Infrastructure as Code** (**IaC**). Terraform allows you to create, deploy, and manage infrastructure across various cloud platforms. Users define and deliver data center infrastructure using a declarative configuration language known as **HashiCorp Configuration Language** (**HCL**), or optionally, **JavaScript Object Notification** (**JSON**).
* **Bicep**: Bicep files let users define infrastructure in Azure in a declarative way. These files can be used multiple times so that resources are deployed in a consistent way. The advantage of Bicep is that it has an easy syntax compared to JSON templates. Bicep addresses Azure Resources directly through **Azure Resource Manager** (**ARM**), whereas in JSON, these resources must first be defined. Quickstart templates for Bicep are available through <https://github.com/Azure/azure-quickstart-templates/tree/master/quickstarts>.
* **CloudFormation**: What Bicep does for Azure, CloudFormation does for AWS. It provisions IaC to AWS, using CloudFormation templates. CloudFormation templates are available on Github: <https://github.com/awslabs/aws-cloudformation-templates>.

All these technologies evolve at the speed of lightning. The challenge that every company faces is how to keep up with all the developments. We will try to give some guidance in the next section.

**Summary**

The goal of this chapter was to provide some common understanding of different cloud concepts and how companies could use these to get the best-of-breed solutions to improve the business. Starting a multi-cloud journey requires proper preparation to get the best out of cloud technology, including emerging technologies such as micro-services, containers, and serverless. We noticed that it can become very complex. We must make sure that we keep the platforms consistent, by decreasing complexity and enabling effective management of workloads. Next, we started our journey by creating a plan for transition and transformation, starting with connectivity, and defining landing zones.

Cloud technology evolves at an extremely high pace. It’s hard to keep up with all the new developments and the release of new features. In the last section, we learned how to stay in control with technology mapping, using the principles of the North Star architecture and technology roadmaps.

With a common understanding of the cloud and the underlying technology, we are ready to start using the cloud. In the next chapter, we will further elaborate on defining our cloud migration strategy by designing service models with the guidance of the Cloud Adoption Framework and tools that are provided through the Well-Architected Framework. We will learn how to set up a service catalog to get the best out of the cloud.

**Questions**

1. What is a CMP?
2. This chapter discussed various Kubernetes deployments in public clouds. Name the managed Kubernetes services of Azure, AWS, and GCP.
3. What are Ignite and Re:Invent?
4. True or false: A North Star is a detailed enterprise architecture.

**Join us on Discord!**

Read this book alongside other users, cloud experts, authors, and like-minded professionals.Ask questions, provide solutions to other readers, chat with the authors via. Ask Me Anything sessions and much more.

Scan the QR code or visit the link to join the community now.

<https://packt.link/cloudanddevops>