**5**

**Cloud Migration and Hybrid Cloud Architecture Design**

Today's organizations need to be more agile to respond to customer demands, which requires the ability to quickly scale up to millions of customers and scale down as needed without impacting the budget. Organizations need to continuously acquire new customers, delighting them while working in a fiercely competitive environment. Cloud migration could be the answer for achieving agility and speed. The cloud enables frequent application releases and reduces costs by applying automation and data center consolidation.

So far, you have learned about various aspects of solution architecture, architecture attributes, and architecting principles. Now everyone is talking about the cloud and organizations are looking to move their workloads into the cloud to optimize operational costs. Public clouds such as **Amazon Web Services** (**AWS**), **Microsoft Azure**, and **Google Cloud Platform** (**GCP**) are becoming the primary destinations to host applications, so it's important to learn about the proposition and methods to migrate to the cloud. In this chapter, you will learn about the various aspects of the cloud and develop cloud thinking, which will also help you understand the upcoming chapters better.

As you learned in *Chapter 1*, *The Meaning of Solution Architecture*, cloud computing refers to the on-demand delivery of IT resources over the web, and you pay as you utilize resources. The public cloud helps you acquire technologies such as compute, storage, networks, and databases on an as-needed basis, instead of buying and maintaining your own data centers.

With cloud computing, the cloud vendor manages and maintains the technology infrastructure in a secure environment and organizations access these resources over the web to develop and run their applications. An IT resource's capacity can go up or down instantly and organizations only pay for what they use.

Now, the cloud is becoming essential for every enterprise strategy. Almost every organization decreases its spending by moving into the public cloud, and on top of saving costs, they convert upfront capital expenditure into operational expenditure. A lot of startups born in the last decade started in the cloud and were fueled by *cloud infrastructure* for rapid growth. As enterprises move to the cloud, they must focus on cloud migration strategy and hybrid cloud.

In this chapter, you will learn about the various strategies of cloud migration and hybrid cloud by covering the following topics:

* Benefits of cloud native architecture
* Creating a cloud migration strategy
* Choosing a cloud strategy
* Steps for cloud migration
* Creating a hybrid cloud architecture
* Taking a multi-cloud approach
* Designing a cloud native architecture
* Popular public cloud choices

By the end of this chapter, you will have learned about the benefits of the cloud and be able to design cloud native architecture. You will understand different cloud migration strategies and steps. You will also learn about hybrid cloud design and popular public cloud providers.

**Benefits of cloud native architecture**

In recent years, technology has been changing rapidly and new companies have been born in the cloud world, disrupting old and long-standing organizations. Rapid growth is possible because of no upfront cost being involved when organizations use the cloud, and there is less risk in experimentation due to the *pay-as-you-go* model of the cloud compared to paying the upfront cost of hosting your own server.

The *cloud native approach* helps employees in an organization develop innovative thinking and implement their ideas without waiting for the long cycle of infrastructure.

With the cloud, customers don't need to plan excess capacity in advance to handle their peak season, such as the holiday shopping season for retailers; they have the elasticity to provision resources to meet demand instantly. This significantly helps reduce costs and improve the customer's experience. For any organization to stay in the competition, they have to move fast and innovatively.

With the cloud, enterprises are not only able to get their infrastructure quickly across the globe but can also access a wide variety of technologies that were never available before. These include access to cutting edge technologies such as the following:

* Big data and analytics
* Machine learning and artificial intelligence
* Robotics
* **Internet of Things** (**IoT**)
* Blockchain
* Quantum computing

Also, to achieve scalability and elasticity, these are some of the reasons that can trigger an initiative for cloud migration and hybrid cloud strategy:

* The data center needs a technology refresh
* The data center's lease is ending
* The data center has run out of storage and compute capacity
* Modernization of an application
* Leverage cutting-edge technologies
* Need to optimize IT resources to save on operational costs
* Disaster recovery planning and operational resilience
* Utilizing a content distribution network for the website
* Reduce upfront capital expenditures and eliminate maintenance costs
* Increase workforce efficiency and productivity
* Improve business agility

Every organization has a different strategy, and one size does not fit all when it comes to cloud adoption. The frequent use cases are putting development and testing environments in the cloud to add agility for developers so that they can move faster. As hosting web applications is becoming more economical and more straightforward with the cloud, organizations are using the cloud for digital transformation by hosting their websites and digital properties in the cloud.

For application accessibility, it is essential to not only build an application for the web browser but to ensure it is accessible through *smart mobiles* and *tablets*. The cloud is helping with such transformations. Data processing and analytics is another area where enterprises are utilizing the cloud since it is less expensive and faster to collect, store, analyze, and share data with the cloud.

Building a solution architecture for the cloud is slightly different than it is for regular enterprise architecting. While moving to the cloud, you have to develop cloud thinking and understand how to leverage the in-built capabilities of the cloud. For cloud thinking, you follow the *pay-as-you-go* model. You need to make sure that you optimize your workload properly and run your servers only when it's required.

You need to think about how to optimize costs by starting the server for your workload when needed and choosing the right strategy for the workload, which always needs to be running. In the cloud, the solution architect needs to have a holistic view of each component regarding performance, scaling, high availability, disaster recovery, fault tolerance, security, and automation.

The other areas of optimization are **cloud native monitoring** and **alerting mechanisms**. You may not need to bring your existing third-party tool from on-premise to the cloud as you can utilize native cloud monitoring better and get rid of costly third-party licensing software. Also, now, you get to have deployment capabilities to any part of the world in minutes, so don't restrict yourself to a particular region and utilize the global deployment model to build better high-availability and disaster recovery mechanisms.

The cloud provides excellent deals for automation; *you can pretty much automate everything*. Automation not only reduces errors and speeds up time to market; it also saves lots of cost by utilizing human resources efficiently and freeing them up from performing tedious and repetitive tasks. The cloud works on a *shared responsibility model* where cloud vendors are responsible for securing physical infrastructure. However, the security of an application and its data is entirely the customer's responsibility. Therefore, it's important to lock down your environment and keep tabs on security by utilizing cloud native tools for monitoring, alerts, and automation.

Throughout this book, you will learn about the cloud perspective of solution architecture and get an in-depth understanding of cloud architecture. Before defining your cloud strategy, let's learn about some popular public cloud choices that you should know.

**Popular public cloud choices**

Since the cloud is the norm now, there are many cloud providers on the market that provide cutting-edge technology platforms that are competing to get market share.

The following are the major cloud providers (at the time of writing):

* **AWS**: AWS is one of the oldest and largest cloud providers. AWS provides IT resources such as compute power, storage, databases, and other services on a need basis over the internet with a pay-as-you-go model. AWS not only offers IaaS; it has a broad range of offerings in PaaS and SaaS. AWS provides multiple offerings in cutting-edge technologies in the area of machine learning, artificial intelligence, blockchain, **Internet of Things** (**IoT**), and a comprehensive set of significant data capabilities. You can host almost any workload in AWS and combine services to design an optimal solution.
* **Microsoft Azure**: Also known as Azure and like any cloud provider, it provides IT resources such as compute, network, storage, and databases over the internet to its customers. Like AWS, Azure also provides IaaS, PaaS, and SaaS offerings in the cloud, which include a range of services from computing, storage, data management, content distribution networks, containers, big data, machine learning, and IoT. Also, Microsoft has wrapped its popular offerings in the cloud through Microsoft Office, Microsoft Active Directory, Microsoft SharePoint, MS SQL Server, and so on.
* **GCP**: GCP provides cloud offerings in the area of computing, storage, networking, and machine learning. Like AWS and Azure, it has a global network of data centers available as infrastructure as a service for its customers to consume IT resources over the internet. In terms of compute, GCP offers Google Cloud Functions for the serverless environment, which you can compare with AWS Lambda functions in AWS and Azure Functions in Azure. Similarly, GCP offers multiple programming languages for application development with containers so that you can deploy application workloads.

There are many other cloud vendors available, such as Alibaba Cloud, Oracle Cloud, and IBM Cloud, but the major markets are captured by the aforementioned cloud providers. The choice of which cloud provider to use is up to the customer, which can be impacted by the availability of the functionality they are looking for or based on their existing relationship with providers. Sometimes, large enterprises choose a multi-cloud strategy to utilize the best providers. In the next section, you will learn about various strategies for cloud migration.

**Creating a cloud migration strategy**

As we mentioned in the previous section, there could be various reasons for cloud migration, and those play an essential role in your cloud journey. Your cloud strategy helps you to determine a migration strategy and prioritize applications.

In addition to primary business drivers for cloud migration, you could have more reasons related to the data center, business, application, team, and workload for cloud migration.

Cloud adoption is not just about choosing the platform, security design, and operation, but you also need to consider people, processes, and culture in addition to technology. For cloud migration success, you first need to align leaders and earn commitment from teams by upskilling them. You need to define the vision across the organization to ensure a successful cloud transition.

Often, migration projects adopt multiple strategies and utilize different tools accordingly. The migration strategy will influence the time it takes to migrate and how the applications are grouped for the migration process. The following diagram shows some of the commonly used strategies for migrating existing applications to the cloud:

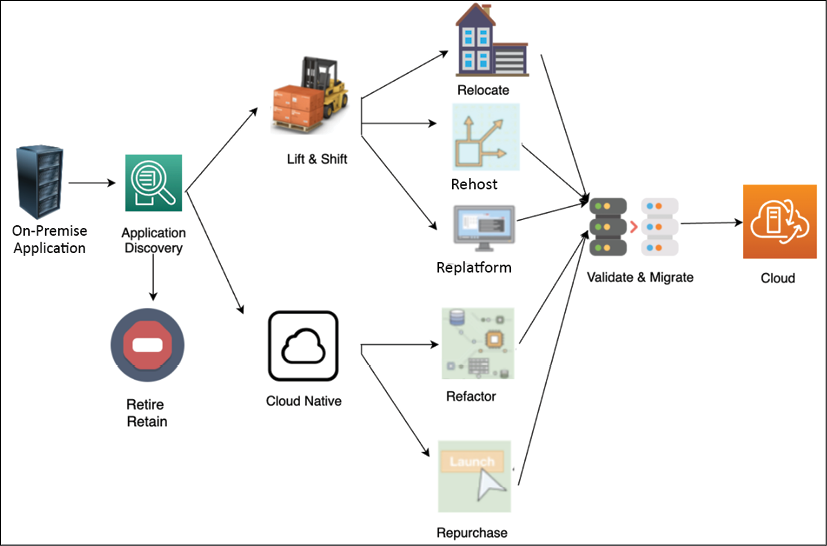


Figure 5.1: Cloud migration strategy

As shown in the preceding diagram, you can do a **Lift & Shift** of the server or application from the source environment to the cloud. Migrating a resource only needs minimal changes for it to work in the cloud. To take a more **Cloud Native** approach, you can refactor your application to fully utilize the cloud native feature, for example, converting monolithic applications into microservices.

If your application is a legacy application and cannot be moved, or it is not cloud compatible, you may want to retire it and replace it with a cloud native SaaS product or third-party solution.

An organization can take a mix of migration strategies; for example, if an application-hosted OS is at its end of life, then you need to upgrade the OS. You can take this opportunity to migrate to the cloud for better flexibility. In this case, most likely, you will choose the **replatform** method to recompile your code into a new version of the OS and validate all its features. After you've finished testing, you can migrate the application to the OS hosted in the infrastructure provided by the cloud. If you want to buy a new platform, for example, replacing your old CRM solution with SaaS-based solutions provided by Salesforce, you can choose a **retire and repurchase** strategy. If you want to rebuild your application from monolithic to microservices to add more agility, you may decide to **refactor**.

Your business objectives will drive your decision to migrate applications and define the strategy for migration as per their priority. For example, when cost efficiency is the main driver, the migration strategy typically involves mass migration with a heavy focus on the **Lift and Shift** approach. However, if the main goal is to enable agility and innovation, the cloud native approach (such as rearchitecting and refactoring) plays a key role in the cloud migration strategy. Let's learn learn more about each strategy in the following subsections.

**Lift and Shift migration**

**Lift and Shift** is the fastest mode of migration, as you need minimal work to move your application. However, it does not take advantage of the cloud native features. Let's look at the most common migration strategies, that is, **rehost**, **replatform**, and **relocate**, which are often utilized to do lift and shift with minimal changes needing to be made to the application.

**Rehost**

Rehost is fast, predictable, repeatable, and economical, which makes it the most preferred method for migrating to the cloud. Rehost is one of the quickest cloud migration strategies, where the server or application is lifted and shifted from the source on-premises environment to the cloud. Minimal changes may be made to the resources during the migration process.

Customers often use rehost to migrate their applications to the cloud quickly and then focus on optimization when the resources are running in the cloud. This technique allows them to realize the cost benefits of using the cloud.

Customers typically use the rehost technique for the following:

* A temporary development and testing environment
* For when servers are running packaged software, such as SAP and Microsoft SharePoint
* When an application doesn't have an active roadmap

While rehost is intended to be applied to packaged software and helps us move quickly into the cloud, you may need to upgrade underlying application platforms such as operating systems. In such a situation, you can use the replatform approach of cloud migration.

**Replatform**

When an operating system, server, or database version gets to its end of life, then it can trigger a cloud migration project, for example, upgrading the operating system of your web server from Microsoft Windows 2003 to Microsoft Windows 2008/2012/2016 or upgrading your Oracle database engine, and so on. The replatform strategy involves upgrading the platform as a part of the cloud migration project but without changing application architecture. You can decide to update your operating system or application to a newer release as part of the migration.

When using the replatform migration strategy, you may need to reinstall your application on the target environment, which triggers application changes. This requires thorough testing on your application after replatforming to ensure and validate its post-migration operational efficiency.

The following common reasons warrant the use of the replatform technique:

* Changing the operating system from 32-bit to 64-bit
* Changing the database engine
* Updating the latest release of the application
* Upgrading the operating system from Windows 2008 to Windows 2012 or 2019
* Upgrading the Oracle database engine from Oracle 8 to Oracle 19C/21C
* To get the benefits of managed services that are available from cloud vendors such as managed storage, databases, application deployment, and monitoring tools

Replatform helps you advance your application's underlying platform while migrating to the cloud. You can simply relocate your application to the cloud if it was deployed in containers or VMware. Now, let's learn more about the relocate strategy.

**Relocate**

You may deploy your application using containers or VMware appliances in your on-premise data center. You can move such workloads to the cloud using the accelerate migrations strategy known as **relocate**. Relocate helps you move hundreds of applications in days. You can quickly relocate applications based on VMware and container technologies to the cloud with minimal effort and complexity.

The relocation strategy does not require much upfront developer investment or an expensive test schedule since it provides the agility and automation you expect from the cloud. You need to determine existing configurations and use **VMotion** or **Docker** to relocate your servers to the cloud. VMotion is known for live migration. It's a VMware technology that enables a virtual instance to be moved from one physical host server to another without any interruption in the service.

Customers typically use the relocating technique for the following reasons:

* Workloads have been deployed in a container
* Applications have been deployed in VMware appliances

**VMware Cloud** (**VMC**) on AWS not only migrates applications, but it migrates thousands of virtual machines, from individual applications to entire data centers. While migrating your application to the cloud, you may want to take the opportunity to rebuild and rearchitect your entire application to make it more cloud native. The cloud native approach allows you to use the full capability of the cloud. Let's learn more about the cloud native approach.

**The cloud native approach**

When your team decides to move to cloud native, in the short term, it seems like more upfront work and slower migration to the cloud. This is a bit costly, but it pays off in the long term when you start using all the cloud benefits with an agile team to innovate.

You will see a drastic decrease in costs over time with the cloud native approach as you can optimize your workload for the right price while keeping performance intact with the *pay-as-you-go* model. Cloud native includes containerizing your application by rearchitecting it as a microservice or opting for a purely serverless approach.

For your business needs, you may want to replace the entire product with a *ready-to-use* SaaS offering, for example, replacing on-premise sales and HR solutions with Salesforce and Workday SaaS offerings. Let's learn more about the refactor and repurchase methods for the cloud native migration approach.

**Refactor**

The refactor method involves rearchitecting and rewriting an application before migrating it to the cloud to make it a cloud native application. In refactoring, you change the application to a more modular design, such as monolithic to microservice. Cloud native applications are applications that have been designed, architected, and built to perform efficiently in a cloud environment. The benefits of these cloud-inherent capabilities include *scalability*, *security*, *agility*, and *cost-efficiency*. Refactoring to microservices helps organizations to create small independent teams that can take complete ownership, thus increase the speed of innovation.

Refactoring requires more time and resources to recode the application and rearchitecture it before it can be migrated. This approach is commonly used by organizations that have extensive cloud experience or a highly skilled workforce. An alternative option for refactoring is to migrate your application to the cloud and then optimize it. You can use cloud native serverless technologies to reduce the admin overhead that comes with modular design.

Common examples of refactoring include the following:

* Changing platforms, such as AIX to UNIX
* Database transition from traditional to cloud databases
* Replacing middleware products
* Rearchitecting the application from monolithic to microservice
* Rebuilding application architecture such as containerizing or making it serverless
* Recoding application components
* Data warehouse modernization to connect organizations to customers

Sometimes, you may find a large effort being made to rebuild an application. As an architect, you should evaluate if purchasing the SaaS product helps you get a better **return on investment** (**ROI**). Let's explore the repurchase strategy in more detail.

**Repurchase**

When your IT resources and projects are migrated to the cloud, you may need servers or applications, which require you to purchase a cloud-compatible license or release. For example, the current on-premises license for your application might not be valid when you run the application in the cloud.

There are multiple ways to address such scenarios of licensing. You can purchase a new license and continue to use your application in the cloud, or you can drop the existing application and replace it with another one in the cloud.

This replacement could be a SaaS offering of the same application.

Common examples of repurchase include the following:

* Replacing the application with SaaS such as Salesforce CRM or Workday HR
* Purchasing a cloud-compatible license

The cloud may not be the answer to all of your problems and sometimes, you will find a legacy application that may not benefit from cloud migration or discover rarely used applications that can be retired. Let's learn about the *retain or retire strategy* in more detail.

**Retain or retire**

When you are planning a cloud migration, it may not be necessary to move all applications. You may need to retain some applications due to technology constraints; for instance, there may be legacy applications coupled with an on-premise server that cannot move. On the other hand, you may want to retire some applications and use cloud native capabilities, for example, third-party application monitoring and alerting systems. Let's learn more about the retain or retire strategy.

**Retain**

You might encounter a few applications in your on-premises environment that are essential for your business but are not suitable for migration because of technical reasons, such as the OS/application not being supported on a cloud platform. In such situations, your application cannot be migrated to the cloud, but you can continue running it in your on-premises environment.

For such servers and applications, you may need to perform only the initial analysis to determine their suitability for cloud migration. However, the server or application may still have dependencies with applications that are migrated. Therefore, you may have to maintain the connectivity of these on-premises servers to your cloud environment. You will learn more about on-premises to cloud connectivity in the *Creating a hybrid cloud architecture* section of this chapter.

Some typical workload examples for retention are as follows:

* A legacy application where the customer doesn't see the benefit of moving to the cloud
* The operating system or application support is not available in the cloud, such as AS400 and a mainframe application

You may want to retain complex legacy systems as on-premise and prioritize them so that they can be moved at a later point; however, during discovery, often, organizations find applications that are not in use anymore but are still sitting around and consuming infrastructure space. You can choose to retire such applications. Let's explore more about the retiring strategy.

**Retire**

While migrating to the cloud, you may discover the following:

* Rarely used applications
* Applications consuming an excessive amount of server capacity
* Applications that may not be required due to cloud incompatibility

In such a situation, you may want to retire the existing workload and take a fresh approach, which is more cloud native.

A retirement strategy can be applied to hosts and applications that are soon going to be decommissioned. This can also be applied to unnecessary and redundant hosted applications. Depending on your business needs, such applications can be decommissioned on-premises without even migrating to the cloud. Hosts and applications that are commonly suited for retirement include the following:

* On-premise servers and storage for disaster recovery purposes
* Server consolidation to resolve redundancies
* Duplicate resources due to mergers and acquisitions
* Alternative hosts in a typical high-availability setup
* Third-party licensed tools such as workload monitoring and automation, which is available as in-built capabilities in the cloud

Most migration projects employ multiple strategies, and there are different tools available for each strategy. The migration strategy will influence the time it takes to migrate and how the applications are grouped for the migration process. Cloud migration is a good time to examine your overall inventory and get rid of the ghost server running under the developer's desk and going unaccounted for. In this section, you learned about various cloud strategies. Let's take a quick look at comparing them in the next section.

**Choosing a cloud strategy**

Choosing the right migration strategy for cloud adoption as per your business drivers is critical. It would be best to consider various constraints such as financial, resource, time, and skill. You can compare the effort required for the different strategies covered in the previous section in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| Migration  Strategy | Description | Time and Cost | Optimization  Opportunities |
| Refactor | Re-architect application into more modularized such as monolithic to microservice | https://learning.oreilly.com/api/v2/epubs/urn:orm:book:9781801816618/files/Images/1.png | https://learning.oreilly.com/api/v2/epubs/urn:orm:book:9781801816618/files/Images/2.png |
| Replatform | Migrate application to upgraded platform without changing core architecture, such traditional database to cloud or higher operating system version | https://learning.oreilly.com/api/v2/epubs/urn:orm:book:9781801816618/files/Images/3.png | https://learning.oreilly.com/api/v2/epubs/urn:orm:book:9781801816618/files/Images/4.png |
| Repurchase | Replacing your current enviroment by purchasing a cloud-based solution | https://learning.oreilly.com/api/v2/epubs/urn:orm:book:9781801816618/files/Images/5.png | https://learning.oreilly.com/api/v2/epubs/urn:orm:book:9781801816618/files/Images/6.png |
| Rehost | Quickly lift and shift your applications to the cloud without architecture changes | https://learning.oreilly.com/api/v2/epubs/urn:orm:book:9781801816618/files/Images/7.png | https://learning.oreilly.com/api/v2/epubs/urn:orm:book:9781801816618/files/Images/8.png |
| Retain | Leaving the application on-premises for now at least | https://learning.oreilly.com/api/v2/epubs/urn:orm:book:9781801816618/files/Images/9.png | NA |
| Relocate | Quickly relocate applications to the cloud without changing them, such as container-based applications | https://learning.oreilly.com/api/v2/epubs/urn:orm:book:9781801816618/files/Images/9.png | NA |
| Retire | Identify the assests that are no longer useful and remove them entirely | NA | NA |

To reduce the cloud migration risk, it's always recommended to take a phased approach when migrating applications to the cloud. First, prioritize business functionality and then optimize applications to realize the difference in cost-saving, performance improvement, and resource productivity. Try to migrate first, and in subsequent phases, you can go for optimization. For example, if you are migrating an application that uses an MS SQL database and replacing it with a cloud native database such as Amazon Aurora, the best approach is to migrate the application in the first phase, followed by migrating the database while monitoring risk and application stability in the second phase. You can choose to optimize your application in subsequent steps by using a cloud native serverless tech stack such as AWS Lambda and Amazon DynamoDB.

The migration strategy should be defined so that it can be executed quickly by allowing teams to work independently. The cloud migration strategy can impact other organizational factors such as building engineering functions within the organization rather than outsourcing. You can build a DevOps culture in the organization by automating the entire code testing and deployment pipeline.

Often, the customer sees the typically unseen advantage of optimizing the workload and tightening security while running application discovery to prepare for migration. There are multiple phases involved in cloud migration. In the next section, you will learn about the steps for cloud migration.

**Steps for cloud migration**

In the previous section, you learned about different migration strategies and grouped your application in order to apply the appropriate migration technique. These strategies are also known as the 7 R's (retain, retire, relocate, rehost, repurchase, replatform, and refactor), and some or all of them could be part of your cloud journey.

Since you may need to perform and manage multiple applications in the cloud, it's better to set up a cloud **Center of Excellence** (**CoE**) and standardize this process with a cloud migration factory. The cloud CoE includes experienced people from various IT and business teams across the organization that act as a dedicated cloud team focused on accelerating the building of cloud expertise in the organization. The cloud migration factory defines migration processes and tools, as well as the steps that need to be taken, as shown in the following diagram:

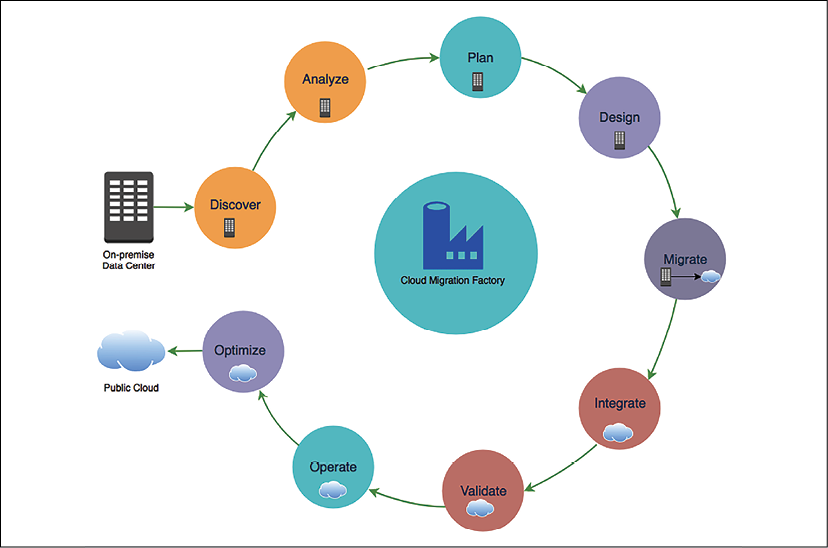


Figure 5.2: Cloud migration steps

As shown in the previous diagram, the cloud migration steps include the following:

* **Discover**: Discovery of cloud migration portfolios and on-premise workloads
* **Analyze**: Analyze discovered data and workloads
* **Plan**: Plan migration to the cloud and define the migration strategy
* **Design**: Design the application as per the migration strategy
* **Migrate**: Execute the migration strategy
* **Integrate**: Integrate with dependencies
* **Validate**: Validate functionality after migration
* **Operate**: Plan to operate in the cloud
* **Optimize**: Optimize your workload for the cloud

One of the initial steps of a cloud migration project is to assess and prioritize the applications for migration. To accomplish this, you need to get a complete inventory of the IT assets in your environment to determine which servers, applications, and business units are suitable for migrating to the cloud, prioritize the migration plan, and determine a migration strategy for these applications. Let's drill down into each step and learn more about them.

**Discovering your workload**

In the discovery phase of your migration project, you discover and capture detailed data about your *cloud migration portfolio*, for example, the scope of your migration project. You identify servers and applications in your portfolio, as well as their interdependencies and current baseline performance metrics. In addition to that, workload discovery includes understanding the existing storage, such as the database and file system, networking configurations, security and compliance needs, application release frequency, DevOps model, escalation path, operating system maintenance and patching, licensing requirements, as well as other associated assets.

Then, you analyze the gathered information to determine application connectivity and capacity requirements, which can guide you in designing and architecting the target cloud environment and identifying the cost. To consider all factors you need to have cross-functional discussions with other business units, from IT to marketing and program management, which helps in aligning changes to support cloud migration processes.

Detailed discovery can also help in identifying any issues in the current state of the application that might need mitigation before you migrate to the cloud. While analyzing the discovery data, you will also determine an appropriate migration method for your application.

**Portfolio discovery** is the process of identifying all the IT assets that are involved in your cloud migration project, including servers and applications, their dependencies, and performance metrics.

You will also need to gather business details about your resources, such as the **Net Present Value** (**NPV**) of the resource, the refresh cycle of the application, the roadmap of the application, and the business criticality of the server or application. These details will help you determine your migration strategy and create a migration plan. In most organizations, these details are maintained across multiple business units and teams. Therefore, during the process of discovery, you may have to interact with various teams, such as business, development, data center, network, and finance.

It is essential to understand that your discovery landscape will depend on various factors:

* What has already been migrated to the cloud?
* What application dependencies are there, along with resources and assets?
* What are the business drivers for cloud migration?
* What is the estimated duration for the entire migration project?
* How many phases is the migration process going to happen in?

One of the top challenges of a migration project is *determining interdependencies among applications*, particularly since they pertain to **input/output** (**I/O**) operations and communications. Cloud migration becomes even more challenging as organizations expand due to mergers, acquisitions, and growth. Organizations often do not have complete information about the following:

* The inventory of the number of servers
* Server specifications such as the type and version of OS, RAM, CPU, and disk
* Server utilization and performance metrics
* Server dependencies
* Overall networking details

Performing thorough portfolio discovery helps in answering questions such as the following:

* Which applications, business units, and data centers are good candidates for migration?
* How suitable are the applications for migrating to the cloud?
* What known or unknown risks are associated with migrating an application to the cloud?
* How should the applications be prioritized for migration?
* Which other IT assets is the application dependent on?
* What are the best migration strategies for the application?
* Is it better to have some downtime for the application than to perform a live migration due to its dependencies and risks?

Several tools are available in the market that can help automate the discovery process and provide more detailed information in a variety of formats. These tools can be classified based on various characteristics, such as deployment type, operation, support, and type of data discovered and reported.

Most of the available solutions can be broadly classified into two categories:

* **Agent-based solutions**: They require their software client to be installed on a server to gather the necessary details.
* **Agentless solutions**: They may be able to capture this information without any additional installations.

Some solutions perform *port scanning* to probe a server or host for open ports, while others perform *packet scanning*, which often involves capturing and analyzing network packets to decode the information. The tools also vary based on the granularity of the data that's discovered, the storage types, and the reporting options. For example, some tools can provide a higher stack of intelligence beyond the network and can also determine the type of applications running.

The complexity of the discovery process depends on the organization's workload and if it already has a well-maintained inventory in place. Discovery processes are typically run for at least a couple of weeks to gather more holistic information about your environment. Once you discover all the necessary information, you need to analyze it. Let's look at the analysis step in more detail.

**Analyzing the information**

To identify server and application dependencies, you need to analyze the network connectivity data, port connections, system, and process information on the hosts. Depending on your tool, you can visualize all the contacts from a server to identify its dependencies, or you can run queries to list all the servers running a specific process, using a particular port, or talking to a specific host.

To group your servers and applications for migration scheduling, you need to identify patterns in your host configurations. Often, some prefixes are embedded in the server hostnames to signify their association with a particular workload, business unit, application, or requirement. Some environments might also use tags and other metadata to associate such details with the host.

To right-size your target environment, you can analyze the performance metrics for your servers and applications:

* If a server is *over-provisioned*, you can revise your right-size mapping information. You can also optimize this process by leveraging the utilization data for the server/application instead of the server specifications.
* If a server is *under-provisioned*, you might assign a higher priority to the server to migrate to the cloud.

Depending on the environment, the type of data that's captured during the discovery process might vary. The data analyzed for migration planning is to determine target network details such as firewall configuration, workload distribution, and the phases in which the application will be migrated.

You can combine this insight with the availability of your resources and business requirements to prioritize your cloud migration workload. This insight can help you in determining the number of servers to be included as part of each cloud migration sprint.

Based on the discovery and analysis of your cloud migration portfolio, you can determine an appropriate cloud migration strategy for your applications. For instance, servers and applications that are less complex and run on a supported OS might be suitable candidates for a lift and shift strategy. Servers or applications that run on an unsupported OS might need further analysis to determine an appropriate strategy.

In a cloud migration project, discovery, analysis, and planning are tightly integrated. You perform a full discovery of your cloud migration portfolio and analyze the data to create a migration plan. By the end of the analysis phase, based on your analysis and the details you've gathered from business owners, you should be able to do the following for each server/application that is part of your cloud migration portfolio:

* Choose a migration strategy for the server/application, depending on your organization's cloud adoption strategy. You may be limited to specific choices within retain, retire, relocate, repurchase, rehost, replatform, and refactor.
* Assign a priority for migrating the resources to the cloud. Eventually, all the resources that are part of the cloud migration portfolio may migrate to the cloud, but this priority will determine the urgency of that migration. A higher-priority resource might move earlier in the migration schedule.
* Document the business driver for migrating the resources to the cloud, which will drive the need and priority for migrating the resources to the cloud.

Planning utilizes the information collected in the discovery and analysis phase to create migration waves. Waves are logical groupings of resources that can be sequentially deployed into production and test/dev environments during cloud migration. Let's look at migration planning in more detail.

**Creating a migration plan**

The next phase in your migration project is *planning cloud migration*. You will use the information you gathered during the portfolio discovery phase to create an efficient migration plan. By the end of this phase in your migration project, you should be able to create an ordered backlog of applications that can migrate to the cloud.

The main goals of the migration planning phase include the following:

* Choosing a migration strategy
* Defining the success criteria for the migration
* Determining the right size of the resources in the cloud
* Determining a priority for applications to migrate to the cloud
* Identifying migration patterns
* Creating a detailed migration plan, checklist, and schedule
* Creating migration sprint teams
* Identifying tools for migration

In preparation for the migration planning phase, you must perform a detailed discovery of all the IT assets that are part of your cloud migration portfolio. The target destination environment in the cloud is also architected before the planning phase. Migration planning includes determining the cloud account structure and creating a network structure for your application. It is also essential to understand hybrid connectivity with the target cloud environment. Hybrid connectivity will help you plan for applications that might have dependencies on resources that are still running on-premise.

The order of application migration can be determined through three high-level steps:

1. Evaluate each application across several business and technical dimensions associated with a potential migration to accurately quantify the environment.
2. Identify the dependencies for each application with qualifications such as locked, tightly coupled, and loosely coupled to identify any dependency-based ordering requirements.
3. Determine the desired prioritization strategy of the organization to determine the appropriate relative weighting of the various dimensions.

The initiation of an application or server migration depends on two factors:

* First, the prioritization strategy of your organization and the application priority. Your organization might place varying emphasis on a few dimensions, such as maximizing ROI, minimizing risk, ease of migration, or another custom dimension.
* Second, the insight gained through the portfolio discovery and analysis phase can help you identify application patterns that match its strategy.

For example, if the organizational strategy is to minimize the risk, then business criticality will have more weight in identifying the applications. If ease of migration is the strategy, applications that can be migrated using rehost will have higher priority, as rehost is a more straightforward process than other strategies. The outcome of planning should be an ordered list of applications that can be used to schedule the cloud migration.

The following are the planning aspects of migration:

1. Gather baseline performance metrics for your applications before migration. Performance metrics will help you design or optimize your application architecture in the cloud quantitatively. You might have captured most of these performance details during the discovery phase.
2. Create test plans and user acceptance plans for your applications. These plans will help in determining the outcome (success or failure) of the migration process.
3. You also need to have cutover strategies and rollback plans that define how and where the applications will continue to run based on the outcome of the migration.
4. Operations and management plans will be useful for determining the ownership of roles during migration and post-migration. You can leverage **Responsible, Accountable, Consult, Inform** (**RACI**) matrix spreadsheets to define these roles and responsibilities for your application that span the entire cloud migration journey.
5. Identify points of contact within the application team that can provide timely support in case of escalations. Close collaboration across the teams will ensure the successful completion of the migration as per the schedule (sprint).

If your organization has some of these processes already documented for your existing on-premises environment, for example, change control process, test plans, and run books for operations and management, you might be able to leverage them.

You need to compare performance and cost before, during, and after migration, which can be an indication that they are not currently capturing enough of the right **Key Performance Indicators** (**KPIs**) to enable this insight. The customer needs to identify and begin achieving useful KPIs so that there is a baseline to compare against during and after migration. The KPI approach in migration has a twofold goal. First, it needs to define the capabilities of your existing application and then compare them with the cloud infrastructure.

When the new products are added to the catalog or a new service is launched, it increases your company revenue, and that's a count against company KPIs. Generally, IT metrics include the quality of the product and the number of bugs that are reported for an application. A **Service-Level Agreement** (**SLA**) defined for fixing a critical bug, system downtime, and performance metrics includes system resource utilization values such as memory utilization, CPU utilization, disk utilization, and network utilization.

You can use a continuous delivery methodology such as **Scrum** to efficiently migrate applications to the cloud. With the help of the Scrum methodology, you can create multiple sprints and add your applications to the sprint backlogs based on prioritization. You can sometimes combine many applications and create waves that follow a similar migration strategy and are possibly related to each other. Typically, you would maintain a constant duration across sprints and vary the application based on factors such as sprint team size and the complexity of the application.

If you have small teams that have in-depth knowledge about the applications that need to be migrated, then you can use weekly sprints, where each sprint consists of the discover/analyze, plan/design, and migrate phases, with a final cutover on the last day of the sprint. However, as the team iterates through the sprints, the workload in each sprint can increase because the teams have now gained experience in the migration process and can incorporate the feedback from previous sprints to make the current sprint more efficient with continuous learning and adaptation.

If you are migrating a complex application, you could also use the entire week for just the plan/design phase and perform the other phases in separate sprints. Tasks that you perform within the sprint and their deliverables can vary, depending on factors such as complexity and team size. The key is to get value from the sprint.

You can create multiple teams to assist in the migration process, depending on various factors such as your product backlog, migration strategy, and organizational structure. Some customers create groups focused on each migration strategy such as a rehost team, a refactor team, and a replatform team. You could also have a team specialized in optimizing your application architecture in the cloud. The multi-team strategy is the preferred model for organizations that have a large number of applications to be migrated to the cloud.

The team can be divided into the following segments:

* First, the team can validate the essential components to ensure your environment (dev, test, or prod) is working, adequately maintained, and monitored.
* The integration team will determine the application configuration and also find the dependencies, which will help reduce the waste that's made by another team.
* The lift and shift migration sprint team migrates large applications that don't require refactoring or replatforming. The team will use automation tools to deliver small amounts of incremental value after every sprint.
* The replatform migration sprint team focuses on application architecture changes in order to migrate applications to the cloud, for example, modernizing application design for microservices or updating the operating system to the latest version.
* The refactor migration sprint team is responsible for managing various migration environments such as production, testing, and development. They make sure all the environments are scalable and functioning as required by monitoring them closely.
* The innovation migration sprint team works collaboratively with groups such as the foundation and transition team to develop a package solution that can be used by other groups.

It's recommended that you run a pilot migration project while planning and continuously building a product backlog so that these adaptations and lessons learned can be incorporated into the new plan. It's best to target non-production migration waves first in the pilot phase. The successful results of the pilot project and sprint can also be used to help secure stakeholder buy-in for the cloud transformation program.

**Designing the application**

During the design phase, your focus should be on successfully migrating applications and making sure your application design meets the required success criteria and is up to date after it has been migrated to the cloud. For example, if you are maintaining user sessions in the on-premise application server (so that it can scale horizontally), make sure that a similar architecture is implemented in the cloud after the migration, which defines the success criteria.

It is essential to understand that the primary goal of this phase is to ensure that your application has been designed to meet the migration success criteria. You need to identify opportunities that enhance your application, and they can be accomplished and achieved during the optimization phase.

For migration, first, you need to have a complete understanding of your organization's foundational architecture on-premises and in the cloud, which includes the following:

* User account
* Network configuration
* Network connectivity
* Security
* Governance
* Monitoring

Knowledge of these components will help you to create and maintain a new architecture for your application. For example, if your application handles sensitive information such as **Personally Identifiable Information** (**PII**) and has control access, this means your architecture needs a specific network setting to meet compliance needs.

During the design phase, you will identify the architecture gap and enhance your architecture as per your application requirements. When you have multiple accounts, each account may have some level of relationship or dependency; for example, you can have a security account to ensure that all your resources are compliant with company-wide security guidelines.

When thinking about your application's network design, you need to consider the following:

* Network packet flows entering the boundaries of your application
* External and internal traffic routing
* Firewall rules for network protection
* Application isolation from the internet and other internal applications
* Overall network compliance and governance
* Network log and flow audit
* Separation of application risk levels, as per their exposure to data and users
* DDoS attack protection and prevention
* Network requirements for production and non-production environments
* SaaS-based multi-tenancy application access requirements
* Network boundaries at the business unit level in an organization
* Billing and implementation of the shared services model across the business unit

You can consider hybrid connectivity options with an on-premise system, depending on your connectivity needs. To build and maintain a secure, reliable, performant, and cost-optimized architecture in the cloud, you need to apply best practices. Review your cloud foundational architecture against the cloud best practices before migrating to the cloud.

*Chapter 4*, *Principles of Solution Architecture Design*, highlights common architectural design patterns that you can consider when migrating your application to the cloud. It is important to emphasize here that the primary goal of the design phase in the migration process is to design your application architecture so that it meets the migration success criteria identified in the planning phase. Your application can be further optimized during the optimization phase of the migration project.

In the process of migrating to the cloud, you can design your application architecture so that it benefits from the global cloud infrastructure and increases the proximity to your end users, mitigates risk, improves security, and addresses data residency constraints. Systems that are expected to grow over time should be built on top of a scalable architecture that can support growth in users, traffic, or data with no drop in performance.

For applications that need to maintain some state information, you could make specific components of the architecture stateless. If there are any layers in the architecture that need to be stateful, you could leverage techniques such as session affinity to be still able to scale such components. Leverage a distributed processing approach for applications that process vast amounts of data.

Another approach to reducing the operational complexity of running applications is using serverless architectures. These architectures can also reduce costs because you are neither paying for underutilized servers nor provisioning redundant infrastructure to implement high availability. You will learn more about serverless architecture in *Chapter 6*, *Solution Architecture Design Patterns*.

The following diagram shows a migration design from on-premises to the AWS cloud, starting with the on-premises design:

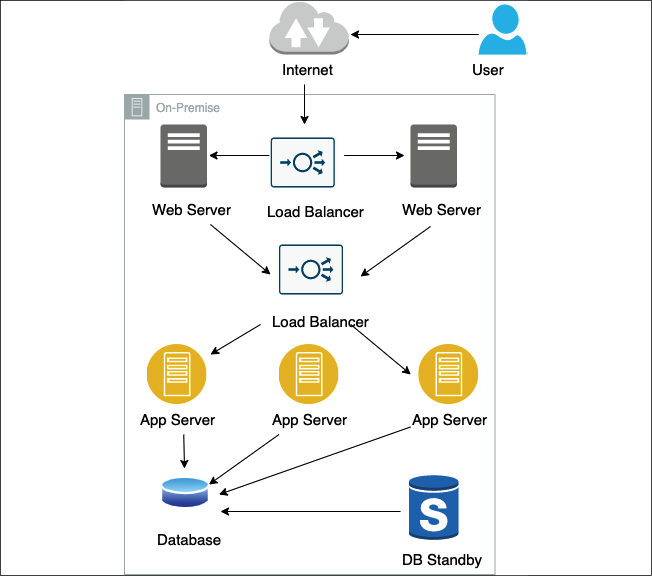


Figure 5.3: On-premise architecture mapping

Now we transition to an AWS cloud design:

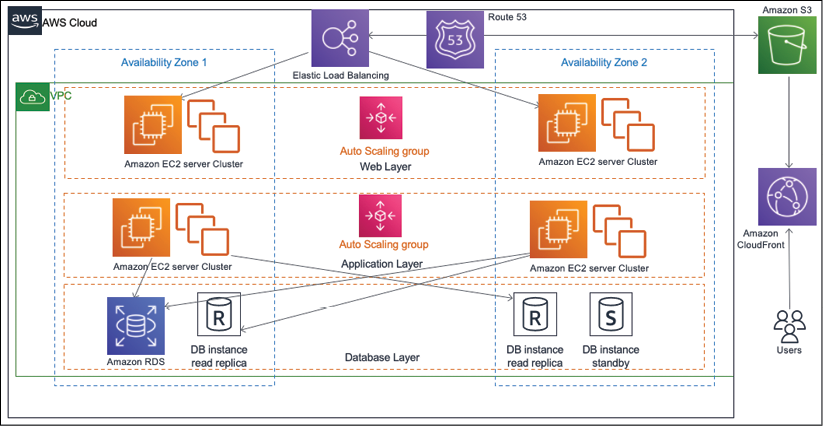


Figure 5.4: On-premise to AWS cloud architecture mapping

In the preceding diagram, as part of the cloud migration strategy, it was determined to rehost the web servers and introduce auto-scaling to provide the elasticity that can help meet spikes in demand. Elastic load balancers are also added to distribute the incoming traffic to the web server instances. The application servers were migrated using refactor, and the platform for the database tier changed from the traditional database to a cloud native **Amazon RDS**. The entire architecture is distributed across multiple availability zones to provide high availability, and the database replicates to a standby instance in the second availability zone.

As an output of your design phase, you should create a detailed design document for the architecture of your application in the cloud. The design document should include details such as the user account that the application must migrate to, network configuration, and a list of users, groups, and applications that need access to the data held by this application. The design document should clearly articulate application hosting details and application-specific requirements for backup, licensing, monitoring, security, compliance, patching, and maintenance. Ensure that you create a design document for each application. You will need it during the migration validation phase to perform a basic cloud functionality check and an application functionality check.

**Performing application migration to the cloud**

The migration execution step brings your plans to fruition. In the execution phase, you need to define a set of steps and configurations, as you will repeat them during the dev/test and production waves. Before executing migration, ensure that you have a migration plan and that you have identified the sprint teams and migration waves and schedules, have created a prioritized backlog, and have notified all the application stakeholders about the migration schedule, timelines, and their roles and responsibilities.

You must also ensure that the target environment in the cloud has already been set up with the foundational architecture and core services. You might have some application-specific pre-steps, such as performing a backup or sync before migration, shutting down the servers, or unmounting disks and devices from the server. Make sure you put in place your essential components, such as networking and firewall rules, authentication and authorization, and accounts. All need to be configured appropriately. You need to test your applications on the infrastructure to make sure that they have access to required servers, load balancers, databases, authentication servers, and so on. You need to pay special attention to application logging and monitoring to measure performance comparisons.

Make sure you have good network connectivity with the cloud environment during the migration process. A good estimate of the amount of data that needs to be migrated also helps you properly estimate the time it will take to migrate your data to the cloud, given other factors such as bandwidth and network connectivity. You also need to understand the tools that are available to perform the migration. Given the number of devices that are available in the market, you might have to narrow down the selection criteria based on your requirements and other constraints.

As you know, rehost is often the fastest way to migrate your application to the cloud. When the application is running in the cloud, you can further optimize it to leverage all the benefits that the cloud has to offer. By quickly migrating your applications to the cloud by applying the lift and shift approach, you may start realizing the cost and agility benefits sooner.

Depending on the migration strategy, you typically migrate the entire server, including the application and the infrastructure that the application is running on, or just the data that belongs to an application. Let's look at how to migrate data and servers.

**Data migration**

Cloud data migration refers to the process of moving existing data to a new cloud storage location. Most applications will require data storage throughout their progression into the cloud. Storage migration typically aligns with one of two approaches, but organizations may perform both at the same time:

* First, a single lift-and-shift move. This may be required before new applications can be started up in the cloud.
* Second, a hybrid model weighted toward the cloud, which results in newly architected cloud native projects with some legacy on-premises data. The legacy data stores may shift toward the cloud over time.

However, your approach to migrating data will vary. It depends on factors such as the amount of data, network and bandwidth constraints, the data classification tier (such as backup data, mission-critical data, data warehouses, or archive data), and the amount of time you can allocate for the migration process.

If you have extensive archives of data or data lakes in a situation where bandwidth and data volumes are unrealistic, you might want to lift and shift the data from its current location straight into a cloud provider's data centers. You can do this either by using dedicated network connections to accelerate network transfers or by physically transferring the data over the hard drive.

If your data stores can gradually migrate over time, or when new data is aggregating from many non-cloud sources, consider methods that provide a friendly interface to the cloud storage service. These migration services can leverage or complement existing installations such as backup and recovery software or a **Storage Area Network** (**SAN**).

For a small-scale database, one-step migration is the best option, which requires you to shut down the application for from a couple of hours to a few days as per the complexity of the workload. During the downtime, all information from the database is extracted and migrated to the destination database in the cloud. Once the database has been migrated, it needs to be validated with the source database for no data loss. After that, a final cutover can be completed.

In the other case, if a system requires minimal downtime, a two-step migration process is more commonly used for databases of any size:

* In the first step, information is extracted from the source database.
* In the next step, data is migrated while the database is still up and running. You can configure **change data capture** (**CDC**) to ensure all data is migrated and the application is in a working state during migration.

In the entire process, there is no downtime. After the migration task has been completed, you can perform functionality and performance tests for connectivity to external applications or any other criteria as needed.

During this time, because the source database is still up and running, changes will need to be propagated or replicated before the final cutover. At this point, you would schedule downtime for the database, usually a few hours, and synchronize the source and destination databases. After all the change data has been transferred to the target database, you should perform data validation to ensure a successful migration and finally route application traffic to a new cloud database.

You might have mission-critical databases that cannot have any downtime. Performing such zero-downtime migrations requires detailed planning and the appropriate data replication tools. You will need to use continuous data replication tools for such scenarios. It is important to note here that source database latency can be impacted in the case of synchronous replication as it waits for data to be replicated everywhere before responding to the application while the replication is happening.

You can use asynchronous replication if your database downtime is only for a few minutes. With zero-downtime migration, you have more flexibility regarding when to perform the cutover since the source and target databases are always in sync.

**Server migration**

There are several methods you can use to migrate a server to the cloud:

* The host or **OS cloning** technique involves installing an agent on the source system that will clone the OS image of the system. A snapshot is created on the source system and then sent to the target system. This type of cloning is used for a one-time migration. With the **OS Copy** method, all operating system files are copied from the source machine and hosted on a cloud instance. For the OS copy method to be effective, the people and/or tool that executes the migration must understand the underlying OS environment.
* The **disaster recovery** replication technique deploys an agent on the source system that's used to replicate data to the target. However, the data is replicated at the file system or block level. A few solutions continuously replicate the data to target volumes, offering a continuous data replication solution. With the **Disk Copy** method, the disk volume is copied in its entirety. Once the disk volume has been captured, it can be loaded into the cloud as volumes, which can then be attached to a cloud instance.
* For virtual machines, you could use agentless techniques to export/import your VM into the cloud. With the **VM Copy** method, the on-premise virtual machine image is copied. If the on-premise servers are running as virtual machines, such as VMware or OpenStack, then you can copy the VM image and import it into the cloud as a *machine image*. One main benefit of this technique is that you can have server backup images that can be launched over and over again.
* With the **User Data Copy** method, only the application's user data is copied. Once the data has been exported from the original server, you can choose one of three migration strategies—*repurchase, replatform, or refactor*. The user data copy method is only viable for those who know the application's internals. However, because it only extracts user data, the user data copy method is an OS-agnostic technique.
* You can containerize your application and then redeploy it in the cloud. With the containerization method, both the application binary and user data are copied. Once the application binary and user data have been copied, it can be run on a container runtime that is hosted on the cloud. Because the underlying platform is different, this is an example of the replatform migration strategy.

Several migration tools in the market can help you migrate your data and/or server to the cloud. Each major public cloud provides its own tool for migration; however, you can also use other popular cloud migration tools such as CloudEndure, NetApp, Dynatrace, Carbonite, Microfocus, and so on. Some tools take a disaster recovery strategy for migration, and some disaster recovery tools also support continuous replication to facilitate live migrations. There are some that specialize in forklifting your servers, performing database migrations across platforms, or database schema conversion. The tool must be able to support business processes that you are comfortable with, and you must have the operational staff to manage it.

**Integration, validation, and cutover**

Migration, integration, and validation go hand in hand as you want to do continuous validation while performing various integration with your application in the cloud. The team starts by performing the necessary cloud functionality checks to ensure that the application is running with proper network configuration (in the desired geolocation) with some designated traffic flow. Instances can start or stop as desired when the basic cloud functionality check is complete. You need to validate that the server configuration (such as RAM, CPU, and hard disk) is the same as intended.

Some knowledge of the application and its functionality is required to perform these checks. When the primary check is complete, then you can perform integration tests for the application.

These integration tests include checking integration with external dependencies and applications; for example, to make sure the application can connect to Active Directory, **Customer Relationship Management** (**CRM**), patch or configuration management servers, and shared services. When integration validation is successful, the application is ready for cutover.

During the integration phase, you integrate the application and migrate it to the cloud with external dependencies to validate its functionality. For example, your application might have to communicate with an Active Directory server, a configuration management server, or shared services resources that are all external to the application. Your application may also need to be integrated with external applications that belong to your clients or vendors, such as a supplier receiving a feed from your APIs after a purchase order placement.

When the integration process is complete, you need to validate the integration by performing unit tests, smoke tests, and **user acceptance tests** (**UATs**). The results from these tests help you get approval from the application and business owners. The final step of the integration and validation phase includes a sign-off process from the application and business owner of the application, which will allow you to cut over the application from on-premises to the cloud.

The final phase of the cloud migration factory is the **cutover process**. In this phase, you take the necessary steps to redirect your application traffic from the source on-premise environment to the target cloud environment. Depending on the type of data or server migration (one-step, two-step, or zero-downtime migration), the steps in your cutover process may vary. Some factors to consider when determining a cutover strategy include the following:

* Acceptable downtime for the application
* The frequency of the data update
* Data access patterns such as read-only or static data
* Application-specific requirements such as database syncs, backups, and DNS name resolutions
* Business constraints, such as the day or time during which the cutover can happen and the criticality of the data
* Changing management guidelines and approvals

Live migration is most popular for business-critical workload migration. Let's learn more about it.

**Live migration cutover**

The following diagram illustrates a cutover strategy for live zero-downtime migration. In this method, the data is continuously replicated to the destination, and you perform most of the functional validation and integration testing at the destination while the application is still up and running:

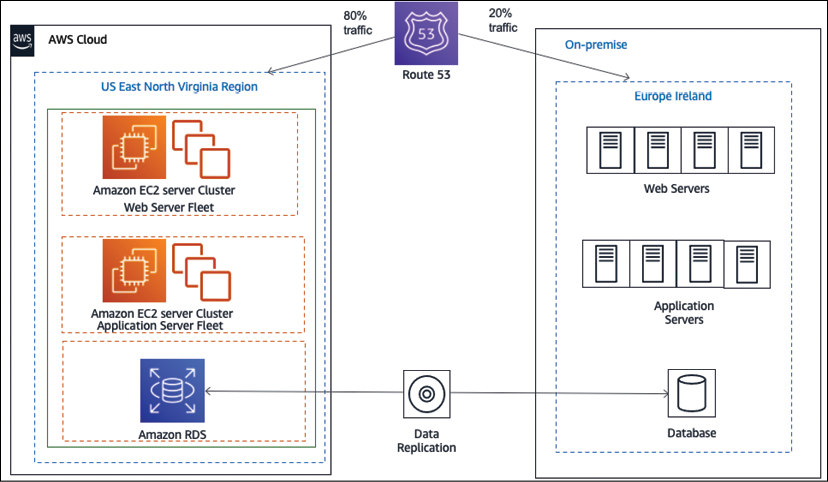


Figure 5.5: Live migration cutover using blue-green deployment

In the replication process, the source on-premise database and target cloud database are always in sync. When all the integration and validation tests are completed successfully and the application is ready for cutover, you can take a **blue-green approach** to do the cutover. The idea behind blue-green deployment is that your blue environment is your existing production environment carrying live traffic. In parallel, you provision a green environment, which is identical to the blue environment other than the new version of your code. You will learn more about blue-green deployments in *Chapter 12*, *DevOps and Solution Architecture Framework*.

Initially, the application continues to run both on-premises and in the cloud, resulting in traffic being distributed between the two sides. You can increase traffic to cloud applications gradually until all the traffic is directed to the new application, thus resulting in a cutover with no downtime.

The other most commonly used cutover strategies involve some downtime. You schedule downtime for the application, pause the traffic, take the application offline, and perform the final sync by applying the CDC process.

After the final sync, it might be a good idea to perform a quick smoke test on the destination side. At this point, you can redirect the traffic from the source to the application running in the cloud, thus completing the cutover. Data is most critical to sync and cutover during migration as it changes continuously when an application is live. You can use data migration tools such as AWS **Database Migration Service** (**DMS**) and Oracle GoldenGate to perform one-time data migration of CDC data.

**Operating the cloud application**

The operation phase of the migration process helps you to allow, run, use, and operate applications in the cloud to the level agreed upon with the business stakeholders. Most organizations typically already have guidelines defined for their on-premises environments. This operational excellence procedure will help you identify the process changes and training that will allow operations to support the goals of cloud adoption.

Let's discuss the differences between deploying complex computing systems in a data center versus deploying them in the cloud. In a data center environment, the burden of building out the physical infrastructure for a project falls on the company's IT department. This means you need to make sure that you have the appropriate physical environmental safeguards for your servers, such as power and cooling, so that you can physically safeguard these assets, and that you have maintained multiple redundant facilities at various locations to reduce the chances of a disaster.

The downside of the data center approach is that it requires significant investment; it can be challenging to secure the resources that are necessary if you wish to experiment with new systems and solutions.

In a cloud computing environment, this changes dramatically. Instead of your company owning the physical data center, the physical data center is managed by the cloud provider. When you want to provision a new server, you ask your cloud provider for a new server with a certain amount of memory, disk space, data I/O throughput rate, processor capability, and so on. In other words, computing resources become a service that you can provision, and de-provision as needed.

The following are the IT operations that you would want to address in the cloud:

* Server patching
* Service and application logging
* Cloud monitoring
* Event management
* Cloud security operations
* Configuration management
* Cloud asset management
* Change management
* Business continuity with disaster recovery and high availability

IT organizations typically follow standards such as **Information Technology Infrastructure Library** (**ITIL**) and **Information Technology Service Management** (**ITSM**) for most of these operations. ITSM organizes and describes the activities and processes involved in planning, creating, managing, and supporting IT services, while ITIL applies best practices to implement ITSM. You need to modernize your ITSM practices so that they can take advantage of the agility, security, and cost benefits provided by the cloud.

In traditional environments, the development team and the IT operations team work in their silos. The development team gathers the requirements from business owners and develops builds. System administrators are solely responsible for operations and for meeting uptime requirements. These teams generally do not have any direct communication during the development life cycle, and each team rarely understands the processes and requirements of the other team. Each team has its own set of tools, processes, and approaches, which often leads to redundant and sometimes conflicting efforts.

In a **DevOps** (short for **development and operations**) approach, both the development team and the operations team work collaboratively during the build and deployment phases of the software development life cycle, sharing responsibilities, and providing continuous feedback. DevOps is a methodology that promotes collaboration and coordination between developers and operational teams to deliver products or services continuously. The software builds are tested frequently throughout the build phase in production-like environments, which allows for the early detection of defects or bugs.

This approach is beneficial in organizations where the teams rely on multiple applications, tools, technologies, platforms, databases, devices, and so on in the process of developing or delivering a product or service. You will learn more about DevOps in *Chapter 12*, *DevOps and Solution Architecture Framework*.

**Application optimization in the cloud**

Optimization is a very important aspect of operating in the cloud, and this is a continuous process of improvement. In this section, you will learn about the various optimization areas. There are chapters dedicated to each optimization consideration in this book. The following are the major optimization areas:

* **Performance**: Optimize for performance to ensure that a system is architected to deliver efficient performance for a set of resources, such as instances, storage, databases, and space/time. You will learn more about architecture performance considerations in *Chapter 7*, *Performance Considerations*.
* **Security**: Continuously review and improve security policies and processes for the organization to protect data and assets in the AWS cloud. You will learn more about architecture security considerations in *Chapter 8*, *Security Considerations*.
* **Reliability**: Optimize applications for reliability to achieve high availability and defined downtime thresholds for applications, which will aid in recovering from failures, handling increased demand, and mitigating disruptions over time. You will learn more about architecture reliability considerations in *Chapter 9*, *Architectural Reliability Considerations*.
* **Operational excellence**: Optimize operational efficiency and the ability to run and monitor systems to deliver business value and to improve supporting processes and procedures continually. You will learn more about architecture operational considerations in *Chapter 10*, *Operational Excellence Considerations*.
* **Cost**: Optimize the cost efficiency of an application or a group of applications, while considering fluctuating resource needs. You will learn more about architecture cost considerations in *Chapter 11*, *Cost Considerations*.

As a quick overview of some of the major elements to consider, to optimize costs, you need to understand what is currently being deployed in your cloud environment and the price of each of those resources. By using detailed billing reports and enabling billing alerts, you can proactively monitor your costs in the cloud.

Remember that, in the public cloud, you pay for what you use. Therefore, you will be able to reduce costs by turning off instances when they are not needed. By automating your instance deployment, you can also tear down and build up the instance entirely as required.

As you offload more, you need to maintain, scale, and pay for less infrastructure. Another way to optimize costs is by designing your architecture for *elasticity*. Make sure you right-size your resources, use auto-scaling, and adjust your utilization based on price and need. For example, it might be more cost-efficient for an application to use more small instances than fewer large instances.

Several application architectural modifications can help you improve the performance of your application. One way to improve the performance of your web servers is to offload your web page through caching. You can write an application that lets you cache images, JavaScript, or even full pages to provide a better experience to your users.

You can design n-tier and service-oriented architectures to scale each layer and module independently, which will help optimize performance. You will learn more about this architecture pattern in *Chapter 6*, *Solution Architecture Design Patterns*.

Customers may want to retain a workload on-premise during cloud migration due to a phased approach or inability to migrate to the cloud due to application complexity or licensing issues. In such scenarios, you need to build a hybrid cloud where the on-premise workload can interact with the cloud workload and exchange information seamlessly. Let's learn more details on creating hybrid cloud architecture.

**Creating hybrid cloud architecture**

The value of the cloud is growing, and many large enterprises are moving their workload to the cloud. However, often, it's not possible to move entirely to the cloud in one day, and for most customers, this is a journey. Those customers seek a hybrid cloud model where they maintain a part of the application in an on-premise environment that needs to communicate with the cloud module.

In a hybrid deployment, you need to establish connectivity between the resources running in the on-premises environment and the cloud environment. The most common method of hybrid deployment is between the cloud and existing on-premises infrastructure to extend and grow an organization's infrastructure into the cloud while connecting cloud resources to the internal system. The common causes of setting up a hybrid cloud may include the following:

* You want to have operating legacy applications in an on-premise environment while you refactor and deploy in the cloud with a blue-green deployment model.
* A legacy application such as a mainframe may not have a compatible cloud option and has to continue running on-premise. You need time to refactor the tech stack.
* You need to keep part of the application on-premise due to compliance requirements.
* To speed up migration, keep the database on-premise and move the application server to the cloud.
* The customer wants to have more granular control of part of the application.
* Data ingestion in the cloud from on-premise for the cloud's **Extract, Transform, Load** (**ETL**) pipeline.

Public cloud vendors provide a mechanism for integrations between a customer's existing infrastructure and the cloud so that customers can easily use the cloud as a seamless extension of their current infrastructure investments. These hybrid architecture functionalities allow customers to do everything, from integrating networking, security, and access control to powering automated workload migrations and controlling the cloud from their on-premises infrastructure management tools.

Taking the example of the AWS cloud, you can establish a secure connection to the AWS cloud using a VPN. Since a VPN connection is set up over the internet, there may be latency issues due to multiple router hops from third-party internet providers. You can have your fiber optics private line go to the AWS cloud for better latency using AWS Direct Connect.

As shown in the following diagram, with AWS Direct Connect, you can establish high-speed connectivity between your data center and the AWS cloud to achieve a low-latency hybrid deployment:

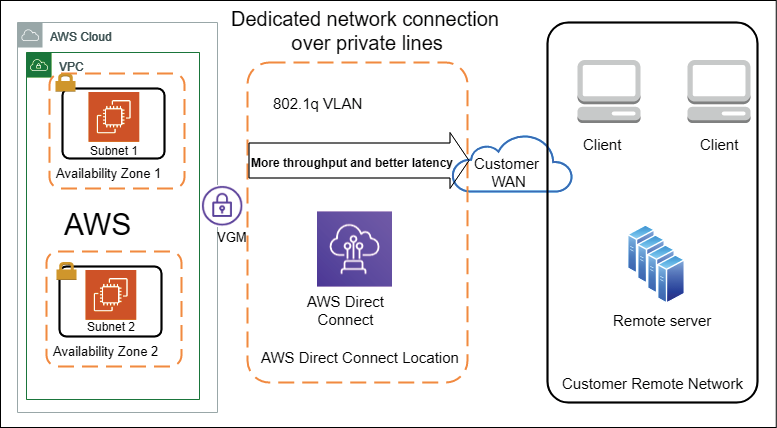


Figure 5.6: Hybrid cloud architecture (on-premises to cloud connectivity)

As shown in the preceding diagram, **AWS Direct Connect Location** establishes the connectivity between the on-premises data center and the AWS cloud. This helps you achieve the customer need of having dedicated fiber-optic lines to an AWS Direct Connect location; the customer can opt for this fiber optic line from a third-party vendor such as AT&T, Verizon, T-Mobile or Comcast in the USA. AWS has a directly connected partner in each region of the world.

At the AWS Direct Connect location, the customer's fiber-optic line is connected to an AWS private network, which provides dedicated end-to-end connectivity from the data center to the AWS cloud. These optic lines can provide speeds of up to 10 GB/s. To secure traffic over direct connect, you can set up a VPN, which will apply IPSec encryption to the traffic flow. As more cloud offerings become available in the market from prominent vendors, organizations may choose to take a multi-cloud approach. Let's learn more details about the multi-cloud strategy.

**Taking a multi-cloud approach**

Before the cloud existed, organizations used multiple vendors to use the best of the breed and avoid vendor lock-in. As more public cloud players come into the market, organizations are looking to create a multi-cloud approach. A multi-cloud approach is about utilizing two or more public cloud providers to serve organization infrastructure and technology needs. The multi-cloud strategy could be a mix of major public cloud providers such as AWS, GCP, Microsoft Azure, Oracle Cloud, IBM, and so on. Organizations can choose to share their workload between different clouds based on their geographical availability, technical capabilities, and cost. They can also combine multi-cloud with on-premise.

One of the major advantages of a multi-cloud strategy is having vendor flexibility. With multi-cloud, you get the advantage of choosing between vendors and retain your negotiation power, agility, and flexibility. In the event of a missed SLA, you have the option to switch over to a better cloud provider. Another advantage is being able to plan disaster recovery in the same region when one cloud provider has an outage; you can rely on other providers. Each cloud provider has its strength, and you can pick the services best available across the cloud.

While the multi-cloud approach provides a competitive advantage to organizations, it also comes with its challenges. One of the most prominent challenges is skill set. You need to have people who understand multiple clouds while creating a workload hosting strategy and, more than that, need to replicate teams to dive deep into each cloud tech stack. You could consider hiring a consultant or outsourcing your cloud management to global system integrators who have a pool of human resources across the cloud.

The other major challenge is coordinating data availability, security, and performance across multiple clouds.

While each cloud vendor provides in-built security, cross-region applications, and cloud native tools for performance, this area pretty much become the organization's responsibility when it comes to the cloud. You need to implement consistent data management across the cloud, taking data from one cloud and feeding it to another, and ensuring consistent performance.

As you can see, the multi-cloud approach has its advantages and disadvantages, so you need to think when choosing a multi-cloud strategy. Once you have started your cloud journey, you may want to build cloud native applications. Let's learn more about building cloud native architecture.

**Designing cloud native architecture**

You learned about the cloud native approach earlier in this chapter from a migration point of view, where the focus was on refactoring and rearchitecting applications when migrating to the cloud. Each organization may have a different opinion on cloud native architecture but, at the center of it, becoming cloud native is all about utilizing all the cloud capabilities in the best way possible. True cloud native architecture is about designing your application so that it can be built in the cloud from its foundations.

Cloud native doesn't mean hosting your application on the cloud platform; it's about leveraging services and features provided by the cloud. This may include the following:

* Containerizing your monolithic architecture in a microservice and creating a CI/CD pipeline for automated deployment.
* Building a serverless application with technology such as AWS Lambda **Function as a Service** (**FaaS**) and Amazon DynamoDB (a managed NoSQL database in the cloud).
* Creating a serverless data lake using Amazon S3 (a managed object storage service), AWS Glue (a managed Spark cluster for ETL), and Amazon Athena (a managed Presto cluster for ad hoc queries).
* Using a cloud native monitoring and logging service, for example, Amazon CloudWatch.
* Using a cloud native auditing service, for example, AWS CloudTrail.

The following architecture is an example of a cloud native serverless architecture for a micro-blogging application:

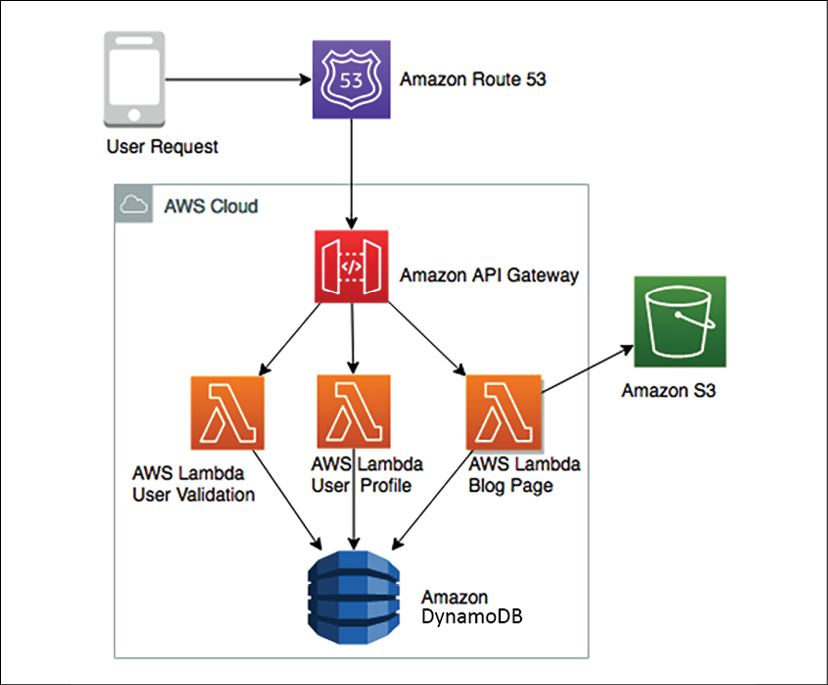


Figure 5.7: Cloud native micro-blogging application architecture

The preceding diagram depicts utilizing cloud native serverless services in the AWS cloud. Here, Amazon Route 53, which manages the DNS service, is routing user requests. Lambda manages function as a service to handle the code for **User Validation**, **User Profile**, and **Blog Page**. All the blog assets are stored in Amazon S3, which manages object storage services, and all user profile data is stored in Amazon DynamoDB, which is managed by the NoSQL store.

As users send requests, AWS Lambda validates the user and looks at their profile to ensure they have a subscription in Amazon DynamoDB; after that, it picks blog assets such as pictures, videos, and a static HTML writeup from Amazon S3 and displays them to the user. This architecture can be scaled in an unlimited manner as all services are cloud native managed services, and you are not handling any infrastructure.

Crucial factors such as high availability, disaster recovery, and scalability are taken care of by these cloud native services so that you can focus on your feature development. In terms of cost, you will only pay if a request goes to a blogging application. If no one is browsing for the blog at night, you don't pay anything for hosting your code; you only pay a nominal fee for storage.

The benefit of cloud native architecture is that it enables fast-paced innovation and agility in the team. It simplifies building out a complex application and infrastructure. As system administrators and developers, you focus strictly on designing and building your networks, servers, file storage, and other computing resources, and leave the physical implementation to your cloud computing provider. Cloud native architecture provides several benefits:

* **Fast scale-out, on-demand**: You can request the resources you need when you need them. You only pay for what you use.
* **Replicate quickly**: Infrastructure-as-code means you can build once and replicate more. Instead of building your infrastructure by hand, you can structure it as a series of scripts or applications. Building your infrastructure programmatically gives you the ability to build and rebuild it on demand, when needed for development or testing.
* **Tear up and tear down easily**: In the cloud, services are provided on-demand, so it's easy to build up a large experimental system. Your system may include a cluster of scalable web and application servers, multiple databases, terabytes of capacity, workflow applications, and monitoring. You can tear it all down as soon as the experiment is completed and save costs.

There are many more examples in the area of storage, networking, and automation for building cloud native architecture. You will learn more about this architecture in *Chapter 6*, *Solution Architecture Design Patterns*.

**Summary**

In this chapter, you learned how the cloud is becoming the most popular mainstream application hosting and development environment for enterprises. At the beginning of this chapter, you learned about cloud thinking and how it's related to solution architecture design. Since more organizations are looking to move into the cloud, this chapter focused on various cloud migration strategies, techniques, and steps.

You learned about various cloud strategies, as per the nature of workload and migration priorities. Migration strategies include the ability to rehost and replatform your application for Lift and Shift and take the cloud native approach by refactoring and rearchitecting your application to take advantage of cloud native capabilities.

You may find some unused inventory during application discovery and retire it. If you choose to not migrate a certain workload, then retain the application as is on-premises.

Then, you learned about the steps involved in cloud migration, which help you discover your on-premise workload, analyze collected data, and create a plan to decide on which migration strategy to take. During the design phase, you create a detailed implementation plan and execute that during the migration steps, where you learned to set up connectivity with the cloud and move your application from on-premise to the cloud.

After that, you learned about how to integrate, validate, and operate your workload into the cloud after migration and apply continuous optimization for cost, security, reliability, performance, and operational excellence. The hybrid cloud architecture is an integral part of the migration process, so you learned how to establish connectivity between on-premise and the cloud by looking at an architecture example of the AWS cloud. At the end of this chapter, you learned about significant cloud providers and their offerings.

In the next chapter, you will dive deep and learn more about various architecture design patterns, along with the reference architecture. You will learn about architecture patterns such as multi-tier, service-oriented, serverless, and microservices.

**Further reading**

To learn more about the major public cloud providers, please refer to the following links:

* **Amazon Web Services** (**AWS**): [https://aws.amazon.com](https://aws.amazon.com/)
* **Google Cloud Platform** (**GCP**): [https://cloud.google.com](https://cloud.google.com/)
* **Microsoft Azure**: [https://azure.microsoft.com](https://azure.microsoft.com/)
* **Oracle Cloud Infrastructure** (**OCI**): <https://www.oracle.com/cloud/>
* **Alibaba Cloud:** [https://us.alibabacloud.com](https://us.alibabacloud.com/)
* **IBM Cloud**: <https://www.ibm.com/cloud>

Almost every cloud provider extends their learning credentials to new users, which means you can sign up with your email and try their offerings out before you choose which one to go with.