**3**

**Leveraging the Cloud for Digital Transformation**

AWS has come a long way since Amazon started in 2006 when it offered just two primary services. In this chapter, you will begin by understanding cloud computing models. Further, you will learn the differences between **Software as a Service** (**SaaS**), **Platform as a Service** (**PaaS**), and **Infrastructure as a Service** (**IaaS**) and how AWS complements each model with its services and infrastructure. You will also learn how today’s businesses use AWS to transform their technology infrastructure, operations, and business practices completely.

In this chapter, we will cover the following topics:

* Cloud computing models: PaaS, IaaS, and SaaS
* Cloud migration strategy
* Implementing a digital transformation program
* The **AWS Cloud Adoption Framework** (**AWS CAF**)
* Architectures to provide high availability, reliability, and scalability

Without further ado, let’s get down to business and learn about terms commonly used to specify how much of your infrastructure will live in the cloud versus how much will stay on-premises.

**Cloud computing models**

Cloud computing allows organizations to focus on their core business and leave unwanted work like IT infrastructure capacity planning, procurement, and maintenance to cloud providers.

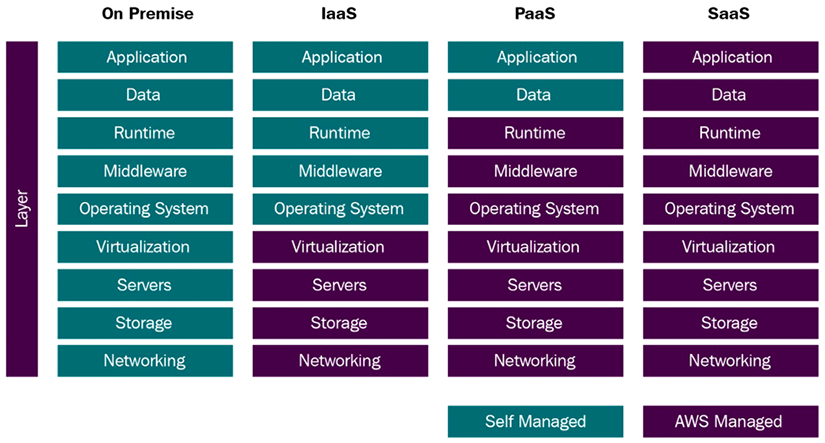
As cloud computing has grown exponentially in recent years, different models and strategies have surfaced to help meet the specific needs of organizations as per their user base. Each type of cloud computing model provides you with additional flexibility and management.

There are many ways to classify cloud services, and understanding the differences between them helps you decide what set of services is suitable for your application workload. In this section, we will cover a common classification. Cloud services can be categorized as follows:

* **Infrastructure as a Service** (**IaaS**)
* **Platform as a Service** (**PaaS**)
* **Software as a Service** (**SaaS**)

As the names indicate, each model provides a service at a different stack level.

Each of these solutions has its advantages and disadvantages. It is essential to fully understand these tradeoffs to select the best option for your organization:



*Figure 3.1: Cloud service classification*

As you can see in the preceding figure, the amount of services managed by yourself or AWS determines how the stack will be classified. On one extreme, we have an on-premises environment, where all the infrastructure is located in your own data center. On the other extreme, we have a SaaS architecture, where all the infrastructure is located on the cloud.

The following sections will explore the advantages and disadvantages of using each and examples of services under each classification.

**Understanding IaaS**

Cloud infrastructure services, also known as **IaaS**, comprise highly flexible, fault-tolerant compute and storage resources. IaaS provides access, complete management, and monitoring of compute, storage, networking, and other miscellaneous services. IaaS enables enterprises to use resources on an *as-needed* basis, so they don’t need to purchase equipment.

IaaS leverages virtualization technology. AWS allows you to quickly provision hardware through the AWS console using the **Command-Line Interface** (**CLI**) or an API, among other methods. By using an IaaS platform, a business can provide a whole host of resources and functionality with its current infrastructure without the headache of physically maintaining it. From a user perspective, they need to be aware that the backend services are being provided by an IaaS platform instead of a company-owned data center.

As you saw in *Figure 3.1*, when using the IaaS solution, you are responsible for managing more aspects of the stack, such as applications, runtime, operating systems, middleware, and data. However, in these cases, AWS will manage servers, hard drives, networking, virtualization, databases, and file storage.

You will now learn the advantages and disadvantages of IaaS services and look at some IaaS use cases; finally, you will see some examples of IaaS offerings.

**Advantages of IaaS**

In the IaaS model, you have maximum control, which means you can achieve the required flexibility as per your application’s needs. These are the advantages of using the IaaS model:

* It offers the most flexibility of all the cloud models.
* Provisioning of compute, storage, and networking resources can be done quickly.
* Resources can be used for a few minutes, hours, or days.
* Complete control of the infrastructure.
* Highly scalable and fault-tolerant.

**Disadvantages of IaaS**

In the IaaS model, having more control means putting additional effort into maintaining, monitoring, and scaling infrastructure. However, there are disadvantages to the IaaS model, some of which, such as data encryption and security issues, vendor lock-in, potential cost overruns, and configuration issues, are also applicable to SaaS and PaaS solutions. More specifically, the disadvantages of IaaS include the following:

* **Security**: In this case, customers have much more control over the stack, and for this reason, it is highly critical that they have a comprehensive plan in place for security. Since customers manage applications, data, middleware, and the operating system, there are possible security threats, such as if certain ports are left open, and intruders guess which ports are open. Attacks from insiders with unauthorized access and system vulnerabilities can expose data between the backend servers and VMs from nefarious sources.
* **Legacy systems**: While customers can migrate legacy applications into AWS, the older hardware may need help to provide the needed functionality to secure the legacy applications. Modifications to older applications may be required, potentially creating new security issues unless the application is thoroughly tested for new security vulnerabilities.
* **Training costs**: As with any new technology, training may be needed for the customer’s staff to get familiar with the new infrastructure. The customer is ultimately responsible for securing their data and resources, computer backups, and business continuity. With this training, it may be easier to secure the necessary staff to support and maintain the new infrastructure.

Let’s look at some use cases where you may want to use the IaaS model.

**Use cases for IaaS**

IaaS is best suited to certain scenarios. This is also true for SaaS and PaaS, as we will see in upcoming sections. IaaS is most suitable when enterprises want more control over their infrastructure. Some of the most common instances when IaaS is used are as follows:

* Backups and snapshots
* Disaster recovery
* Web hosting
* Software development environments
* Data analytics

IaaS provides more flexibility to organizations with granular control; however, that comes with the price of resource management overhead, and you may need help to realize the actual value of the cloud. Let’s look at some examples of IaaS services provided by AWS.

**Examples of AWS IaaS services**

These are a few of the IaaS services offered by AWS:

* **Elastic Compute Cloud** (**EC2**): One of the most popular services in AWS. EC2 is essentially a server on the cloud.
* **Elastic Block Storage** (**EBS**): Amazon EBS is block-level storage. You can think of it as a **SAN** (**Storage Area Network**) drive on the cloud.
* **Elastic File Storage** (**EFS**): Amazon EFS is file-level storage. You can think of it as a **NAS** (**Network Attached Storage**) drive on the cloud.

There are many other AWS IaaS services that you will learn about through this book. Now, let’s learn about **Software as a Service**(**SaaS**).

**Understanding SaaS**

**SaaS**, or *cloud application services*, are services where the cloud provider does most of the heavy lifting (in this case, AWS). As you saw in *Figure 3.1*, you will not have to install software or worry about the operating system or software patches for SaaS. Your focus will be on customizing the application’s business logic and supporting your users. Most SaaS systems will only need browser access as most of the computation will be done on the cloud side.

SaaS eliminates the need for your staff to visit individuals’ devices regarding software installation. Cloud providers such as AWS are fully responsible for any issues on the server, middleware, operating system, and storage levels. Let’s now analyze the characteristics that make up SaaS, the advantages and disadvantages of using a SaaS deployment, and some examples.

**Characteristics of SaaS**

These are the clues that will help determine if a service is SaaS:

* It is managed by the vendor (such as AWS).
* It is hosted on a third-party server.
* It can be accessed over the internet.
* AWS manages applications, infrastructure, operating systems, software patches, and updates.

These characteristics make SaaS unique compared to other models and help organizations offload the burden of infrastructure management to cloud providers such as AWS. Let’s look at some more advantages of the SaaS model.

**Advantages of SaaS**

SaaS has several advantages:

* Reducing the time, money, and effort spent on repetitive tasks
* Shifting the responsibility for installing, patching, configuring, and upgrading software across the service to a third party
* Allowing you to focus on the tasks that require more personalized attention, such as providing customer service to your user base

A SaaS solution allows you to get up and running efficiently. This option, versus the other two solutions, requires the least effort. This option enables companies big and small to launch services quickly and finish a project on time.

**Disadvantages of SaaS**

SaaS solutions have some limitations as well:

* **Interoperability**: Interoperability with other services may be complex. For example, if you need integration with an on-premises application, it may be more complicated to perform this integration. Most likely, your on-premises installation uses a different interface, complicating the integration. Your on-premises environment is an assortment of technology from different vendors, making it challenging to integrate. In contrast, before you spin up your first service, AWS goes to great lengths and performs rigorous testing to ensure that services interoperate and integrate smoothly.
* **Customization**: The convenience of having a vendor such as AWS manage many things for you comes at a price. Opportunities for customization in a SaaS solution will not be as great as with other services that are further down in the stack. For example, an on-premises solution that offers complete control of all levels in the stack will allow full customization. In your on-premises environment, you install the patch if there is a requirement to use a particular version of Linux with a specific security patch. In contrast, installing a particular version of Linux is impossible if you use AWS Lambda as your deployment environment. In fact, with AWS Lambda, the operating system being used under the covers is transparent to you.
* **Lack of control**: If your organization requires that you only use a particular approved version of an operating system, this may not be appropriate. For example, there might be a regulatory requirement requiring detailed testing approval of the underlying operating systems, and if the version is changed, a retest and approval are required. In this case, SaaS will most likely not be an acceptable solution. In a SaaS environment, you have non-deterministic latency issues. In other words, controlling how long your processes will take requires a lot of work.
* **Limited features**: If the SaaS solution you are using does not offer a feature you require, you might only be able to use that feature if the SaaS vendor provides that feature in the future.

**Use cases for SaaS**

SaaS is best suited for scenarios when you want to use out-of-the-box applications without managing application code or IT infrastructure. You may choose a SaaS solution when you don’t see a return on investment for building the platform due to a small number of users, or when you need in-house expertise. Some of the most common instances when SaaS is used are as follows:

* Payroll applications such as ADP
* **Customer Relationship Management** (**CRM**) solutions such as Salesforce
* Workplace collaboration solutions, including Zoom, Cisco Webex, Amazon Chime, Microsoft Teams, Slack, etc.
* Office management solutions such as Microsoft Office 365
* Workspace solutions, including Amazon WorkSpaces, Google Workspace, Microsoft Workspace, etc.

SaaS reduces the risk for organizations as all the application development and maintenance work is offloaded to vendors. Let’s look at some examples of SaaS services provided by AWS.

**Examples of AWS SaaS solutions**

Some of the services that AWS offers that could be classified as SaaS solutions are as follows:

* **Amazon Connect**: Amazon Connect is a cloud-based contact center that offers businesses a cost-effective solution to deliver exceptional customer service across various communication channels with ease. It leverages Amazon’s innovative AI & ML technologies directly in Amazon Connect without having to manage complex integrations. By utilizing Amazon Lex, you can create voice and text chatbots that enhance contact center efficiency. Similarly, Contact Lens for Amazon Connect can aid in comprehending the tone and patterns of customer interactions, while Amazon Connect Wisdom can minimize the amount of time agents spend searching for solutions.
* **Amazon WorkSpaces**: This allows system administrators to provide virtual Microsoft Windows or Linux **Virtual Desktop Infrastructure** (**VDI**) for their users. It obviates the need to purchase, procure, and deploy hardware and eliminates the need to install the software. Administrators can add or remove users as the organization changes. Users can access their VDIs from various supported devices and web browsers. With Amazon WorkSpaces, you no longer have to visit every machine to install commonly used software such as Microsoft Office and other security software. Amazon WorkSpaces enables virtual environments for your users where this software is already installed, and all they need is access to a browser.
* **Amazon QuickSight**: This business intelligence and analytics service creates charts and visualizations, performs ad hoc analysis, and obtains business insights. It seamlessly integrates with other AWS services to automatically discover AWS data sources.
* **Amazon Chime**: Similar to Slack and Zoom, Amazon Chime can be used for online meetings, video conferencing and calls, online chat, and sharing content.

Above are a few examples of SaaS solutions provided by AWS; however, AWS relies on its partner network to build SaaS solutions using AWS-provided services and published in AWS Marketplace for customers to purchase. Let’s look at some third-party SaaS solutions.

**Examples of third-party SaaS solutions**

Many third-party vendors, including some that offer their services on AWS Marketplace, are SaaS solutions. There are multiple examples, but here are a few that decided to build their SaaS offerings on AWS:

* **Splunk**: A software platform that enables search, aggregation, analysis, and visualizations of machine-generated data collected from disparate sources such as websites, mobile apps, sensors, and IoT devices.
* **Sendbird**: A chat solution specializing in real-time chat and messaging development for mobile apps and websites. It provides client-side SDKs in various languages, a user-friendly dashboard, and moderation tools.
* **Twilio**: A company that offers various ways to securely communicate with customers, including email, SMS, fax, voice, chat, and video. Whenever you get an SMS to prove your identity when you log into your bank account, there is a decent chance that Twilio was involved in the process.

This concludes the SaaS section. The following section will cover PaaS, another common paradigm in cloud deployments.

**Understanding PaaS**

Defining a SaaS service is easy. If AWS manages everything, it’s a SaaS service. The same applies to a definition of an on-premises service. If you manage everything on your infrastructure, it’s clear you have an on-premises service. As you begin going up and down the stack and start taking over some of the components’ management or offloading some of the management, the line starts getting fuzzy. We’ll still try to provide you with a definition for PaaS.

An initial definition could be this: any application where you are responsible for the maintenance of some of the software and some of the configuration data. More formally, **Platform as a Service** (**PaaS**) is a cloud computing service that supplies an environment to enable its users to develop, run, and manage data and applications without worrying about the complexity associated with provisioning, configuring, and maintaining the infrastructure. These complexities come in the IaaS model, where you are responsible for creating applications, including the servers, storage, and networking equipment.

As you saw in *Figure 3.1*, PaaS is like SaaS in some ways. Still, instead of providing services to end-users that do not need to be technically savvy to use the software, PaaS delivers a platform for developers to potentially use the PaaS service to develop SaaS solutions.

PaaS enables developers to design and create applications while operating at a very high level of abstraction and focusing primarily on business rules and user requirements. These applications, sometimes called middleware, can be highly scalable and available if developed appropriately. Let’s take an example of a PaaS service called Amazon **Relational Database Service**(**RDS**), where AWS provides managed relational databases such as Oracle, MS SQL, PostgreSQL, and MariaDB in the cloud. AWS handles database engine installation, patching, backup, recovery, repair, etc. You need to build a schema and store your data as per business needs.

Like SaaS, PaaS takes advantage of virtualization technology. Resources can be started or shut down depending on demand. Additionally, AWS offers a wide selection of services to support PaaS applications’ design, development, testing, and deployment, such as AWS Amplify to develop web and mobile apps.

Let’s now look into the advantages and disadvantages of PaaS, some use cases for PaaS, and some examples of services that are considered PaaS services.

**Advantages of PaaS**

It doesn’t matter if you are a three-person start-up or a well-established multinational; using PaaS provides many benefits, such as:

* Cost-effective and continuous development, testing, and deployment of applications, as you don’t need to manage the underlying infrastructure
* High availability and scalability
* Straightforward customization and configuration of an application
* Reduction in development effort and maintenance
* Security policy simplification and automation

Now that you’ve learned about the PaaS model’s pros, let’s consider some cons.

**Disadvantages of PaaS**

PaaS solutions have some limitations as well:

* **Integrations**: Having multiple parties responsible for the technology stack creates complexity in how integrations must be performed when developing applications. That becomes particularly problematic when legacy services are on-premises and are not scheduled to be moved to the cloud soon. One of the reasons enterprises like to minimize the number of technology vendors is not to allow these vendors to be able to point fingers at each other when something goes wrong. When something invariably goes wrong, enterprises know precisely who they must contact to fix the problem.
* **Data security**: The data will reside in a third-party environment when running applications using a PaaS solution. This poses concerns and risks. There might also be regulatory requirements to be met to store data in a third-party environment. Customers might have policies that limit or prohibit the storage of data *off-site*. For example, China recently passed regulations that require **Personally Identifiable Information** (**PII**) generated in China not to leave China. More specifically, if you capture your customer’s email on your site and your site is available in China, the servers that store the email must reside in China, and that email cannot leave the country. Using a PaaS approach to comply with this regulation requires standing up full-fledged infrastructure mimicking your existing infrastructure in other locations.
* **Runtime issues**: PaaS solutions may not support the language and framework that your application may require. For example, if you need an old version of a Java runtime, you might not be able to use it because it may no longer be supported.
* **Legacy system customization**: Existing legacy applications and services might require more integration work. Instead, complex customization and configuration needs to be done for legacy applications to integrate with the PaaS service properly. The result might yield a non-trivial implementation that may minimize the value provided by your PaaS solution. For example, many corporations rely on mainframes for at least some of their needs. If they wanted to move these mainframe applications to the cloud, they would have to rewrite the applications that do not require a mainframe since mainframes are not one of the types of hardware that most typical cloud providers support.
* **Operational limitations**: Even though you have control of some of the layers in the PaaS stack, other layers are controlled and maintained by AWS. If the AWS layers need to be customized, you have little or no control over these optimizations. For example, if you are required to use a particular operating system but your PaaS provider does not support it, you are stuck with choosing one from the list of available operating systems.

Let’s look at when it makes sense to use the PaaS model in more detail.

**PaaS use cases**

PaaS can be beneficial and critical to today’s enterprises’ success. Here are some examples of PaaS use cases:

* **Business Process Management** (**BPM**): Many enterprises use PaaS to enable BPM platforms with other cloud services. BPM software can interoperate with other IT services. These combinations of services enable process management, implementation of business rules, and high-level business functionality.
* **Business Analytics/Intelligence** (**BI**): BI tools delivered via PaaS enable enterprises to visualize and analyze their data, allowing them to find customer patterns and business insights. This enables them to make better business decisions and more accurately predict customer demand, optimize pricing, and determine which products are their best sellers.
* **Internet of Things** (**IoT**): IoT is a key driver for PaaS solution adoption and will likely be even more critical in the coming years. You have only scratched the surface of IoT applications enabled by a PaaS layer.
* **Databases**: A PaaS layer can deliver persistence services. A PaaS database layer can reduce the need for system administrators by providing a fully managed, scalable, and secure environment. You will visit the topic more deeply in a later chapter, but AWS offers a variety of traditional and NoSQL database offerings.
* **API management and development**: An everyday use case for PaaS is to develop, test, manage, and secure APIs and microservices.
* **Master Data Management** (**MDM**): MDM came about from the need of businesses to improve the quality, homogeneity, and consistency of their critical data assets. This critical data includes the customer, product, asset, and vendor data. MDM is used to define and manage this critical data. Additionally, it provides a single point of reference or a *single source of truth* for this data. MDM enables methods for ingesting, consolidating, comparing, aggregating, verifying, storing, and routing essential data across the enterprise while ensuring a common understanding, consistency, accuracy, and quality control. PaaS platforms have proven to be a boon for developing MDM applications, enabling them to process data quickly and efficiently.

Using PaaS is beneficial, sometimes even critical, in many applications. PaaS can streamline a workflow when several parties simultaneously work on the same task. PaaS is functional when customized applications need to be created. PaaS can reduce development and administration costs. Let’s look at some examples of AWS PaaS services.

**Examples of AWS PaaS services**

Here are some examples of the most popular PaaS offerings in the AWS ecosystem:

* **AWS Elastic Beanstalk**: Beanstalk is a simple service that enables the deployment of web applications in various programming languages and can scale up and down automatically.
* **Amazon RDS**: Amazon RDS is another excellent example of a PaaS. Amazon offers a variety of databases, such as MySQL, Postgres, and Oracle. When using Amazon RDS to use these databases, You can focus on writing your applications against them and let Amazon handle the underlying management of the database engine.
* **AWS Lambda**: Lambda is another relatively simple and fully managed service that can quickly scale to handle thousands of requests per second. It requires almost no configuration and removes the worry of providing your hardware. AWS Lambda is called a **Function as a Service** (**FaaS**).
* **Amazon Elastic Kubernetes Service** (**Amazon EKS**): Amazon EKS is a fully managed service that enables running Kubernetes on the cloud without installing Kubernetes or deploying your servers.

Now that we have explored SaaS, PaaS, and IaaS in detail, let’s see when it’s appropriate to use each of them.

**Choosing between SaaS, PaaS, and IaaS**

Each model you learned, including the on-premises model, has advantages and disadvantages. The one you choose depends on your specific business requirements, the features needed, and the developers and testers that comprise your team. You might need an entirely *out-of-the-box* solution, and *time to market* might be a more important consideration than price. Or perhaps you have regulatory constraints that force you to control the environment completely. AWS offers a lot of assurances regarding their **Service Level Agreements** (**SLAs**) and compliance certifications. The more levels in the stack you decide to manage, the more effort you will exert to verify that your systems comply with the different regulations.

In general, one good rule of thumb is to let AWS take over the management of your resources whenever possible. You only take over the responsibility when necessary. For example, imagine trying to implement the functionality that Amazon Elastic Load Balancing or Elastic Kubernetes Service provides.

There are two main reasons why you should use IaaS or PaaS instead of SaaS:

* The use case requires a specific type of database or software not supported by the AWS SaaS solutions. For example, you may already have purchased Tableau licenses for your organization and built reports. So instead of using Amazon QuickSight as a SaaS BI platform, you can install Tableau in EC2 instances as an IaaS model.
* The total cost of ownership of running an application using PaaS or IaaS is significantly lower than the SaaS model. A specific example may be AWS Athena versus using Apache Presto directly. If you plan to run thousands of queries per day, with the current cost structure, some cloud users have found deploying Presto more cost-effective than using AWS Athena. Another option is to use Amazon Redshift as a PaaS model rather than Athena as a SaaS model cost-efficiently. It’s important to note that these cost calculations should be carried out using all relevant costs, including staffing and support costs, not just software costs.

As you’ve learned about different cloud computing models and understand how to make the right choices, let’s learn about cloud migration strategies.

**Cloud migration strategy**

The proportion of IT spending shifting to the cloud is accelerating, with predictions that over 45% of system infrastructure, infrastructure software, application software, and more will move from traditional solutions to the cloud by 2024.

Migrating to the AWS cloud makes your organization more innovative by enabling it to experiment and be agile. The ability to move quickly and achieve business values for your users truly matters for your company’s cloud migration.

By migrating your digital assets to the cloud, you can gain insights from your data, innovate faster, modernize aging infrastructure, scale globally, and restructure organizational models to create better customer experiences. Often, cost reduction is one of the primary drivers of migrating workloads to the cloud. In practice, organizations regularly see the value of migration going well beyond the cost savings from retiring legacy infrastructure.

As you have started to discover, there are additional tasks you can perform in addition to migrating workflows to the cloud. What tasks should be performed and when they should be done will depend on the available budget, staff technical expertise, and leadership buy-in.

It is hard to create discrete cohorts to classify these tasks since they are more of a continuum. Having said this, and without further ado, let’s attempt to create a classification. Keep in mind that this classification is not meant to be dogmatic. You may run into other ways to classify this migration. Additionally, you can mix and match the approaches, depending on your needs. For example, your CRM application may be moved without changing it. But perhaps your accounting software was built in-house, and now you want to use a vendor-enabled solution such as QuickBooks Online.

AWS use a three-phase approach that integrates modernization into the migration transition. AWS’s three-phase process and the seven migration patterns help provide the guiding principles for structuring the cloud migration journey so you can quickly realize continuous, quantifiable business value. Let’s look at this pattern in the three-phase migration process.

**The three-phase migration process**

The AWS cloud migration project typically executes in three phases, from discovery to building cloud readiness and finally migrating to the cloud. The following are three phases of cloud migration:

1. **Assessment phase** – Assessment is the first phase of cloud migration, where you need to conduct a feasibility study of your existing on-premise workload to build cloud readiness. In this phase, you can build cost projections and size your workload to run it in the cloud. The assessment phase helps you to build a business case for cloud migration. AWS acquired TSO logic in 2019 and launched it as the AWS Migration Evaluator, which can gather data from your on-premise workload and provide estimated cost savings in AWS.

You can learn more about migration evaluators by visiting the AWS page here: <https://aws.amazon.com/migration-evaluator/>. It helps you to define a roadmap for AWS migration, look at licensing and server dependencies and generate a **Migration Readiness Assessment** (**MRA**) report. Further, You can use the AWS **Cloud Adoption Framework** (**CAF**) to build cloud readiness and plan your migration strategy. You will learn more details about CAF later in this chapter.

1. **Mobilize phase** – The mobilize phase comes after the assessment phase to address gaps in cloud readiness. In this phase, based on the assessment conducted in the previous phase, you want to build cloud skills in your organization to handle cloud operations. It would be best to build a baseline environment, such as account and user setup, using AWS Landing Zone in order to move your workload. AWS provides the **Cloud Adoption Readiness Tool**, also known as **CART**, which helps you plan for cloud adoption based on your migration readiness. You can learn more about CART from the AWS page: <https://cloudreadiness.amazonaws.com/#/cart>. Further, AWS provides Migration Hub, which facilitates a centralized place for application tracking and migration automation across multiple AWS tools. You can learn more about the AWS migration hub by visiting the AWS page: <https://aws.amazon.com/migration-hub/>.
2. **Migrate and modernize phase** – After building a solid foundation in the mobilize phase, it is time to migrate and modernize. In this phase, you will design, migrate, and validate your workload in the cloud. You can start with creating workstreams such as foundation, governance, and migration to ensure your cloud migration project is operation ready. Further, you can subdivide the migrate phase into two parts, where in the initial phase, you build a runbook for migration and in the implementation phase, you perform the actual migration. Cloud migration allows you to modernize your business by refactoring legacy workloads. However, it may not be necessary to migrate and modernize in one go. Often, the best approach is to move rapidly to the cloud by performing lift and shift; after that, you can modernize by re-architecting the application in AWS and making it more cloud-native.

You can refer to AWS’s prescriptive guidance to learn about the above three-phase cloud migration approach in detail – <https://docs.aws.amazon.com/prescriptive-guidance/latest/large-migration-guide/phases.html>. These phases are standard guidelines for successful cloud migration; however, each organization may have varying needs, and these guidelines are not set in stone.

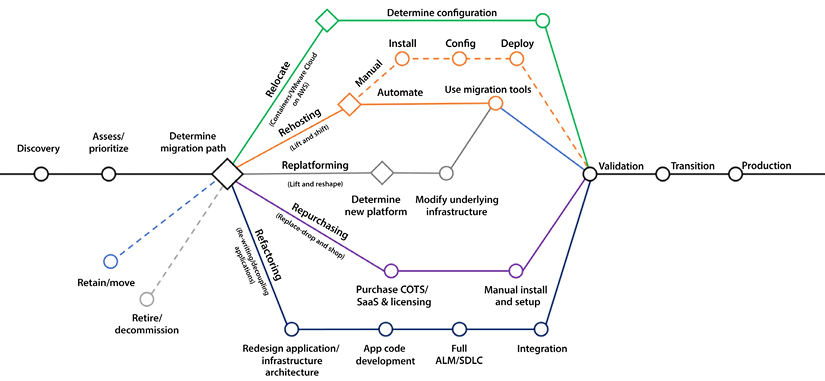
Let’s review the migration patterns, also known as “The 7 Rs,” for migrating to the cloud and learn when to pick one over the others.

**Cloud migration patterns – The 7 Rs**

There is more than one way to handle migration. The following are the 7 Rs of cloud migration patterns defined by AWS:

1. Rehost
2. Re-platform
3. Refactor
4. Relocate
5. Repurchase
6. Retain
7. Retire

Creating a detailed strategy that identifies your workloads’ best patterns is essential to accelerating your cloud journey and achieving your desired business objectives. The following diagram shows the 7 Rs cloud migration model.



*Figure 3.2: 7 Rs of AWS cloud migration*

The diagram above is taken from the AWS blog: <https://aws.amazon.com/blogs/enterprise-strategy/new-possibilities-seven-strategies-to-accelerate-your-application-migration-to-aws/>.

Let’s fully summarize the 7 Rs and what each migration pattern brings to the table.

**Rehost in the cloud**

This method is also commonly known as *lift and shift*. With this method, you perform the least work to move your workloads to the cloud by rehosting your application in the cloud. Applications are rehosted as they are in a different environment. Services are simply migrated. Let’s say, for example, that you are hosting a simple three-tier application on your on-premises environment that is using the following:

* A web server
* An app server
* A database server

Using the *lift and shift* strategy, you would set up three similar servers on the cloud, install the applicable software on each server, and migrate the necessary data. Therefore, this approach will have the lowest migration costs. However, this simplicity comes at a price. Any problems in the existing applications will come along during the migration. If the current applications are obsolete and suboptimal, they will remain obsolete and suboptimal.

Have you ever had to move from one house to another? It’s a painful process. In broad strokes, there are two ways that you can pack for the move:

* You can just put everything in a box and move it to the new house.
* You can be judicious and sort through what you have, item by item, and decide whether you will toss, sell, recycle, or take the thing with you.

Packing everything and going is quick because you avoid sorting through everything, but as you know, it can be more expensive because you will be moving more things, and it is painful because you may realize later that you should not have moved some items.

The move to the cloud is similar. Using the *lift and shift* strategy is good if you are already confident that your processes and workflows are solid and do not need to be changed, but this is rarely the case. This approach takes everything from the on-premise data center and moves it to the cloud. However, additional work is still required, such as preparing servers, creating VPCs, managing user access, and other tasks. You can automate most rehosting with AWS-provided tools such as CloudEndure and Migration Evaluator. AWS Server Migration Service offers agentless capabilities when you cannot install an agent on the server—making it faster and easier for you to migrate large workloads from on-premises to AWS using a snapshot of the existing servers.

The lift and shift approach may not always allow us to optimize the desired cost. However, it is the first step toward the cloud and is still a highly cost-effective option for many organizations. This approach is often the best choice if you want to start the migration process while getting a feel for cloud benefits. Let’s look into other approaches.

**Re-platform in the cloud**

Re-platforming your services to run in the cloud entails migrating the applications and changing the underlying infrastructure architecture. However, the code of the higher-level services will not be changed. This way, you can leverage the existing code base, languages, and frameworks that you are currently using. It may be a good balance between taking advantage of some of the properties of the cloud, such as elasticity and scalability, without making wholesale changes to your existing applications. For example, while migrating to the cloud, you can upgrade Windows-based applications running on an older version, such as \*Windows Server 2008, to the latest version, such as \*Windows Server 2022.

It is advisable to use this method when you are comfortable with your current set of applications but want to take advantage of certain cloud advantages and functionality. For example, you can add failover to your databases without buying the software to run this setup reliably. If you were implementing this functionality *on-premises*, you would have to own all the infrastructure. A specific example is Oracle Data Guard, which allows you to implement this failover but not without having to install the product, and you need enough expertise to ensure that the product is configured correctly. Instead, when you are in a cloud environment, you can leverage the virtualization nature of the cloud, and costs can be shared with other cloud customers.

**Refactor in the cloud**

Refactoring gives you the opportunity to re-architecture your applications. For example, changing your monolithic apps to a more modular microservice-based architecture and making them a cloud-native serverless application. Refactoring is an advanced approach that adds agility and innovation to your business to meet user demand.

The refactoring approach will enable you to use all the cloud features and fully take advantage of them. This will allow you to use state-of-the-art technologies to create new services and reinvent your existing business workflows. There will be considerable work to accomplish this rewrite. In many cases, especially for established enterprises that new start-ups disrupt, they will soon find themselves relegated to a footnote in history if they don’t reinvent themselves.

This approach allows you to make wholesale changes and start from scratch to create your new applications and workflows. It is essential to have subject matter experts and business experts involved in the design process because you may want to change how you do business and suspend all your current beliefs about how things should be done. Overall, migration is a long process and requires the involvement of application developers, business stakeholders, infrastructure engineers, security experts, etc. Collaboration between different teams in organizations is necessary because significant changes to apps and infrastructure are critical for success.

**Revise before migrating to the cloud**

Another potential strategy is to modify, optimize, and enhance the existing applications and code base before migrating to the cloud in preparation for doing so. Only then do you rehost or refactor the applications to the cloud. This may be a good strategy and provide business continuity and enhancement in the long run. The downside of this approach is the cost associated with changing and testing the code upfront. In addition, changing the code in the on-premises environment may only allow you to take advantage of some of the features that creating the code in the cloud would offer. For example, creating reports using AWS QuickSight. AWS QuickSight is an excellent tool for creating dashboards and reports. However, AWS QuickSight can only be used in a cloud environment, not in your on-premises environment, because QuickSight is only supported within AWS.

This method suits you when you know that your applications are suboptimal and need to be revised. You take cloud migration as an opportunity to enhance and fix your applications. Using this approach, you will only need to test your application once. The drawback is that if things go south, it may be challenging to ascertain if the problems that cropped up are because of new bugs in the code or because you migrated to the cloud.

**Repurchase in the cloud**

Repurchase replaces your existing environment and is known as “drop and shop”; you drop legacy applications and purchase more cloud-native software in the repurchase. So with this method, instead of rebuilding your applications, you get rid of them and replace them with commercially available SaaS alternatives such as Salesforce, Workday, ServiceNow, Datadog for Observability, or SAP. Depending on how deep and skilled your talent pool is and their areas of expertise, this option may or may not be more expensive than rebuilding your application.

Using this option, your software costs will likely be higher, but lower development and maintenance costs will be offset. If you decide to rebuild, you will not have to pay for CRM and commercial software licenses, but development cycles will likely be longer. You will have fewer defects, and higher maintenance may apply.

The previous methods of migration implied that all development was done in-house. One difference with the repurchase approach is migrating from in-house systems to software built by professional vendors. As with the other approaches, this approach has advantages and disadvantages. One of the advantages is that the learning curve and the development life cycle will be shortened, where more development will be needed. However, a disadvantage is that the software will require additional licenses and drive the adoption of new solutions that the company may not have used earlier.

**Relocate to the cloud**

The relocation method allows you to move your applications to the cloud without any changes. For example, you can relocate VMware-based on-prem applications to AWS without any changes. It will also help maintain consistent operations between VMware and the AWS cloud.

Overall, the cloud approach, where you focus on a pay-as-you-go model and reduce CapEx cost, will help you to reduce the **Total Cost of Ownership** (**TCO**) as you move to operational costs from upfront capital investment. After moving to the cloud, you can optimize or re-architect to take advantage of the various AWS services. You can add advanced features such as data engineering, machine learning, containerization, and mobility capabilities backed by the power of AWS.

**Retain on-premises**

The retain method means doing nothing for now and leaving your on-premises workload as-is. You may decide to keep your application on-premises due to it being near the end of its life, or because it’s too complex to move now. For example, organizations often decide to retain mainframe application as it has decades of tech debt that no one knows how to migrate, and they need more planning.

Another example is enterprises that want to keep their applications on-premises and near to users due to the need for ultra-low latency or because of compliance regions, especially in the finance industry. In those cases, retaining your application on-premises and working with a hybrid cloud may be the best solution.

**Retire**

Finally, while analyzing your workload for cloud migration, you may realize that many servers are running unutilized, or you have decided to replace the existing application with cloud-native software.

The retire method is to decommission unwanted portions of your IT workload. During the discovery phase of your migration, you may encounter applications no longer being used. By rationalizing your IT portfolio, you can identify assets that are no longer valuable and can be turned off. It will strengthen your business case and direct your team toward maintaining the more widely used resources.

AWS provides prescriptive guidance to plan and decide which migration strategy will fit your workload. You can refer to the AWS guide at <https://docs.aws.amazon.com/prescriptive-guidance/latest/application-portfolio-assessment-guide/prioritization-and-migration-strategy.html> to customize the flow for your enterprise on-premise workload while working on migration planning. Let’s look into some of the tools provided by AWS to help you with cloud migration.

**Migration assessment tools**

You don’t have to reinvent the wheel as you migrate your workloads and projects from your current environment to the cloud. As you can imagine, many others have already started this journey. AWS and third-party vendors offer various tools to facilitate this process. A few examples of services and tools that are worth exploring are as follows:

* **AWS Migration Hub**: AWS Migration Hub is a central repository that can be used to keep track of a migration project.
* **AWS Application Discovery Service**: AWS Application Discovery Service automates the discovery and inventory tracking of different infrastructure resources, such as servers and dependencies.
* **AWS Migration Pattern Library**: This is a collection of migration templates and design patterns that can assist in comparing migration options and alternatives.
* **Cloud Endure Migration**: Cloud Endure Migration is a product offered by AWS that simplifies cloud migration by automating many steps that are necessary to migrate to the cloud.
* **AWS Data Migration Service**: This service can facilitate data migration from your on-premises databases to the cloud, for example, into Amazon RDS.

This is a partial list. Many other AWS and third-party services can assist in your migration. You can find the complete list by visiting the AWS migration page: <https://aws.amazon.com/free/migration/>.

Now that you have reviewed the different ways to migrate to the cloud, let’s understand why you might want to migrate to the cloud to begin with. You will gain this understanding by learning about the concept of digital transformation. Let’s now dive deeper into how organizations leverage digital transformation using the cloud model.

**Implementing a digital transformation program**

You spent some time understanding AWS cloud migration strategies in the previous section. In this section, you will learn how to transform legacy on-premises technologies into the cloud.

As you can imagine, this can be a challenging exercise, especially for large enterprises that have a long history of using old technologies and making significant investments in them. Deciding to start migrating applications and on-premises services to the cloud is a decision that takes time to make. A complete migration will likely take years and potentially cost millions of dollars in migration, transformation, and testing costs.

For this reason, important decisions need to be made along the way. Some of the most critical decisions that need to be made are as follows:

* Should you perform the bare minimum amount of tasks to achieve the migration, or do you want to use this change as an opportunity to refactor, enhance, and optimize our services? Doing the bare minimum (only migrating your workloads to the cloud) will mean that any problems and deficiencies in the current environment will be brought over to the new environment.
* Should the migration be purely technological, or should you use this opportunity to transform current business processes? You could thoroughly assess how your organization does business today and determine how to improve it. This will create efficiencies, cut costs, and increase customer satisfaction. However, this option will inherently have a higher upfront cost and may or may not work.

In this section, you will start learning the primary strategies for migration to the cloud and weigh up some of the options. It explains why and how you may want to undertake a digital transformation and the benefits and pitfalls that can come with this.

**What exactly is a digital transformation?**

The term “digital transformation” is more complex and harder to define because it is overloaded to the point that it has become a nebulous concept. Like many fantastic technology trends, it is over-hyped and over-used.

According to **International Data Corporation** (**IDC**), some studies report that up to 40% of tech spending will be on digital transformation, while enterprises plan to spend over $2.8 trillion by 2025. The source of these details is the following website: <https://www.idc.com/>.

The term “digital transformation” has become something that simply means platform modernization, including migrating on-premises infrastructure to the cloud. You can blame CIOs, consultants, and third-party vendors for this confusion. They are all trying to convince the C-Suite that their solution can cover today’s enterprise infrastructure and business requirements.

But savvy high-level executives understand that there is no magic bullet, and a digital transformation will require planning, strategizing testing, and a great deal of effort to accomplish.

Let’s nail it down and define it.

**DIGITAL TRANSFORMATION DEFINITION:**

Digital transformation involves using the cloud and other advanced technology to create new or change existing business flows. It often involves changing the company culture to adapt to this new business type. The end goal of digital transformation is to enhance the customer experience and to meet ever-changing business and market demand. Now, cloud migration is an essential part of digital transformation.

A digital transformation is an opportunity to reconsider everything, including the following:

* The current structure of teams and departments
* Current business flows
* The way new functionality is developed

For a digital transformation to succeed, it should be broader than one aspect of the business, such as marketing, operations, or finance. It should eventually be all-encompassing and cover the whole gamut of how you engage with your customers. It should be an opportunity to completely transform how you interact with your potential and existing customers. It should go beyond simply swapping one server in one location for another more powerful or cheaper one in the cloud.

In some regards, start-ups have a big advantage over their more significant, established rivals because they don’t have to unlearn and reimagine their processes. Start-ups have a clean slate that can be filled with anything in the AWS service catalog and other technologies. Existing players must wipe the slate clean while keeping their existing client base and finding a way to keep the trains running while performing their digital transformations.

Digital transformation goes well beyond changing an enterprise’s technology infrastructure. For a digital transformation to be successful, it must also involve rethinking processes, using your staff in new ways, and fundamentally changing how business is done.

Disruptive technological change is usually undertaken to pursue new revenue sources or increase profits by creating efficiencies. Today’s customers continue to raise the bar of expectations driven by so many successful businesses that have delivered on the execution of their digital transformations.

In the next section, you will learn about some of the forces that push companies into embarking on digital transformation. The status quo is a powerful state. Most companies will find it challenging to move from what’s already working, even though they may realize that the current approach could be better. It usually takes significant pressure to bite the bullet and migrate to the cloud finally.

**Digital transformation drivers**

One of the most important reasons companies are finally beginning to migrate their workloads to the cloud and transform their business is because they realize if they don’t disrupt themselves, someone else will do it for them. They see competition from start-ups that can start with a clean slate and without legacy baggage, and they also see incumbent competitors embarking on digital transformation initiatives.

Another obvious example is none other than Amazon’s e-commerce operations. In this case, many of its competitors failed to adapt and have been forced to declare bankruptcy. A partial list of famous retailers that had to file for bankruptcy is as follows:

* Tailored Brands
* Lord & Taylor
* Brook Brothers
* Lucky Brand
* GNC
* J.C. Penney
* Neiman Marcus
* J. Crew
* Modell’s Sporting Goods
* Pier 1

Let’s take examples of companies that have survived and thrived by migrating to the cloud or creating their applications in the cloud.

**Digital transformation examples**

Digital transformation without tangible positive business outcomes will inevitably result in a short stay with your employer or the marketplace. Innovation might be refined in academia and research institutions. Still, in the business world, innovation must always be tied to improvement in business metrics such as increased sales or higher profits.

Remember that digital transformation could mean more than just moving your operations to the cloud. As you saw in the previous section, it may involve refactoring and replacing existing processes. Furthermore, it could also mean utilizing other previously unused technologies, such as robotics, the **Internet of Things** (**IoT**), blockchain, and machine learning.

For example, how many restaurants offer advanced booking through the web or via a mobile app?

A few more concrete examples are as follows:

* TGI Fridays use virtual assistants to enable mobile ordering.
* McDonald’s uses voice recognition technology in their drive-throughs, and they have almost replaced their point-of-sale system with a self-ordering kiosk.
* Chipotle restaurants in the US have entirely changed their ordering model during the Covid-19 pandemic. Instead of allowing customers to enter the restaurant and order, customers had to place their orders via the Chipotle mobile app. Customers would get a time when they could come up and pick up their order or, if they ordered far enough in advance, they could choose when to pick it up.
* Rocket Mortgage (previously Quicken Loans) has upended the mortgage industry by enabling consumers to apply for a mortgage in a streamlined manner and without needing to speak to a human. To achieve this, they heavily relied on technology, as you can imagine.

What are some best practices when implementing a digital transformation? In the next section, we will help you navigate so that your digital transformation project is successful, regardless of how complicated it may be.

**Digital transformation tips**

There are many ways to implement a digital transformation. Some methods are better than others. This section will cover some of our suggestions to shorten the implementation time and minimize disruption to your existing customer base. Let’s look at those tips.

**Tip #1 – Ask the right questions**

You should not just be asking this:

* How can we do what we are doing faster and better?

You should also be asking the following:

* How do we change what we are doing to serve our customers better?
* Can we eliminate certain lines of business, departments, or processes?
* What business outcomes do we want to achieve when interfacing with our customers?
* What happens if we do not do anything?
* What are our competitors doing?

Having a precise understanding of your customer’s journey and experience is critical.

**Tip #2 – Get leadership buy-in**

Digital transformations have a much better chance of success when performed from the top down. If there is no buy-in from the CEO and the rest of the C-Suite, cloud adoption is destined to be relegated to a few corners of the enterprise but has no chance of full adoption. This does not mean that a **Proof of Concept** (**POC**) cannot be performed in one department to work out the kinks. Once the technology is adopted in that department, the bugs are worked out, and tangible business results are delivered, we can roll out this solution to all other departments.

**Tip #3 – Delineate objectives**

In this day and age, where Agile development is so prevalent, it is not uncommon to pivot and change direction as new requirements are discovered. However, the overall objective of the digital transformation should be crystal clear. Is the objective to merely lift and shift the current workflows into the cloud? Then keep your eye on the prize and ruthlessly concentrate on that goal. Is the digital transformation supporting a merger between two companies? In that case, completing the union of both companies’ backend systems and operations should take precedence over everything else. Whatever the goal is, you need to focus on completing that objective before taking on other initiatives and transformations.

**Tip #4 – Apply an agile methodology to your digital transformation**

Embrace adaptive and agile design. The days of waiting for a couple of years to start seeing results, only to discover that you were climbing the wrong mountain, are over.

Many corporations now run with lean budgets and only provide additional resources once milestones have been reached and functionality has been delivered. Embracing an adaptive design enables transformation advocates to quickly tweak the transformation strategy and deploy staffing and resources where they can have the highest impact.

There needs to be a healthy push and pull between accomplishing the objectives for the digital transformation and the inevitable changes in how the goals will be met. If some of the objectives change midstream, these changes need to be clearly defined again. Make sure to precisely spell out what is changing, what is new, and what is no longer applicable.

Agile increases ROI by taking advantage of features as soon as they are available instead of waiting for all functionality to be delivered. Adaptability must be deeply ingrained in the ethos and culture of your digital transformation team members.

Look for singles rather than home runs. Home run hitters typically also have a lot of strikeouts. Players that specialize in hitting singles get on base much more often. You should take the same approach in your digital transformation. Instead of attempting a moon shot, taking smaller steps that produce results is highly recommended. If you can demonstrate value early in your transformation, this will validate your approach and show leadership that your approach is working. How much job security do you think you will have if your transformation takes three years and the project falls behind with no tangible results?

Pick the low-hanging fruit and migrate those workloads first. You will be able to provide quick results with this approach and learn from the mistakes you make in the process, which will help you when you need to accomplish other, more difficult migrations.

**Tip #5 – Encourage risk-taking**

In other words, fail fast. There are only so many times you can fail to deliver results. But if failing only takes one week and you have a month to deliver results, that affords us the luxury of failing three times before we get it right the fourth time. Therefore, in the first couple of attempts, you can attempt to shoot further and achieve more. Ideally, you don’t have to completely throw out the work performed in the first few attempts, and you can reuse what was created in the first phases. But at the very least, you can use the lessons learned from those mistakes.

It’s better to disrupt yourself than to have someone do it for you.

**Tip #6 – One-way door vs. two-way door decisions**

One way to define risk strategy is to understand if investing in digital transformation for a project or department is a one-way door or two-way door decision.

A one-way door decision is where once you start, there is no way to go back due to the amount of investment, and a two-way door decision is where you can easily roll back steps and reduce the risk. Try to have more two-way door decisions where you can retract if things go wrong and have fewer one-day door decisions where you have to move forward once started, and there is no looking back.

You can be more agile and fast in two-way door decisions where you define the existing strategy and timeline; however, you need to be extra careful and analyze more data for one-way door decisions. For example, in a two-way door decision, you can move your HR payroll application to migrate to the cloud and keep an exit strategy to purchase SaaS solutions like Workday or ADP if migration is incomplete in two months or after specific budgets. However, if you decide to move your e-commerce application to the cloud, it will impact your end-user experience. Hence, you must carefully analyze data as it will be a one-way door decision, and there is no going back without a significant impact.

**Tip #7 – Clear delineation of roles and responsibilities**

Fully delineate roles and responsibilities. Make sure that all team members are aligned on their responsibilities and check that there are no gaps in your team. Ideally, you will have a good mix of people with vast experience in cloud migration, digital transformation, and process optimization. Couple that with engineers and analysts that are not billing at an expert rate but can execute the plan laid out by these expert resources.

Current technology in general, and AWS in particular, is changing at an ever-increasing pace. Therefore, attracting talent with the right skills is an essential yet difficult step in digital transformation.

Some of the positions that will most likely need to be filled in your journey are as follows:

* Software engineers
* Infrastructure architects
* Cloud computing specialists
* Data analysts and data scientists
* Solution architects
* Security specialists
* Project managers
* Quality assurance testers
* DevOps administrators
* UX designers
* Trainers and documentation specialists
* Business analysts

The above is a partial list of positions, and your project may require more or fewer people to fill these roles. Not all roles will be required. And in your case, you may need additional roles to those included in this list.

This section taught us best practices and what to do in your cloud migration project. In the next section, you will learn what you should not do and how to avoid making mistakes.

**Digital transformation pitfalls**

There are many more ways to fail and not as many ways to succeed. There are, however, common patterns to how digital transformations typically fail. Let’s review some of them.

**Lack of commitment from the C-suite**

Even when the CEO says they are committed to completely transforming their business, they may still clearly delineate a vision and the path to success or fail to provide the necessary resources for the transformation to succeed.

**Not having the right team in place**

It isn’t easy to know what you don’t know because you don’t know it. It may take reading this sentence a couple of times before it can be understood. Still, the important takeaway is that you should engage people that have performed similar digital transformations to the one you are trying to attempt. Why reinvent the wheel if someone else has already invented it?

Many reputable consulting companies specialize in cloud migration and digital transformation. Your chance of success increases exponentially if you engage them to assist you with your initiative. They understand the challenges, and they can help you avoid the pitfalls. AWS has an extensive Partner Network that can help you migrate to the cloud.

These resources may come with a hefty price tag, and engaging them may take work. Many digital transformation initiatives fail because of a failure to engage the people with the right expertise to perform them.

**Internal resistance from the ranks**

With many of these transformations, there may be an adjustment in personnel. Some new people may join the team, in some cases permanently. Some consulting staff may be brought in temporarily, and some staff may become obsolete and need to be phased out. It is essential to consider the friction these changes will create and deal with them accordingly. When moving workloads to the cloud, some of the on-premises administrators may no longer be required, and you can fully expect that they may be a roadblock to the completion of the migration of these workflows.

Specifically, it is common for infrastructure and system administrators to resist cloud adoption. They often sense that some of their responsibilities may disappear or change; in many instances, they are right. Properly communicating the objective with a focus on training, how the migration will occur, and delineating new responsibilities is key to a successful migration.

**Going too fast**

To succeed, you must crawl before you walk and walk before you run. It is essential to prove concepts at a smaller scale before scaling them up across the enterprise and before you spend millions of dollars. Taking this route will allow you to make small mistakes and refine the transformation process before implementing it enterprise-wide. Remember one-way vs. two-way door decisions while investing in cloud migration projects.

A highly recommended method is to perform PoC projects before going all in, not just for cloud migration but for any project in general. For example, if you have 100 databases in your organization, it is okay to migrate only one or a few to the cloud instead of doing all of them simultaneously.

**Going too slow**

Once you prove the process in a couple of guinea pig departments, it is also essential to refrain from implementing the lessons learned one department at a time. Once you have the suitable template, it is recommended to roll out the new technology across the board. Taking a step-by-step approach may need to be faster to enable you to keep up with more nimble competitors.

Once you get familiar with the migration process and absorb the lessons learned from the first migration, you can accelerate your migration and migrate more applications.

**Outdated rules and regulations**

In some cases, the reason for failure may be outside the hands of the company’s leadership. Current regulations may be a stumbling block to success. In this case, business leaders may have made the mistake of thinking they would be able to change the rules and failed, or the rules may have changed in the middle of the game.

Take, for example, the real estate industry in the US. Proving that someone owns a property and recording such ownership requires recording physical documents in the courthouse, in many cases requiring wet signatures. With the advent of blockchain and other enabling concepts, the technology already exists to transform local governments’ archaic and heterogeneous methods. However, a patchwork of laws at the state level and a wide array of methods used to record these documents at the county level are disrupting this industry and preventing this technology from being implemented.

Another example was at an online pharmacy. They wanted a pill-dispensing robot to fill thousands of prescriptions per minute. As you can imagine, this was an expensive machine costing millions of dollars. However, the company had a restriction: they had to be faxed for medicine prescriptions to be filled, and many customers had trouble faxing in their prescriptions. Hence, the robot pill dispenser ended up being heavily underutilized. Other reasons contributed to its demise, but unfortunately, this enterprise eventually went under.

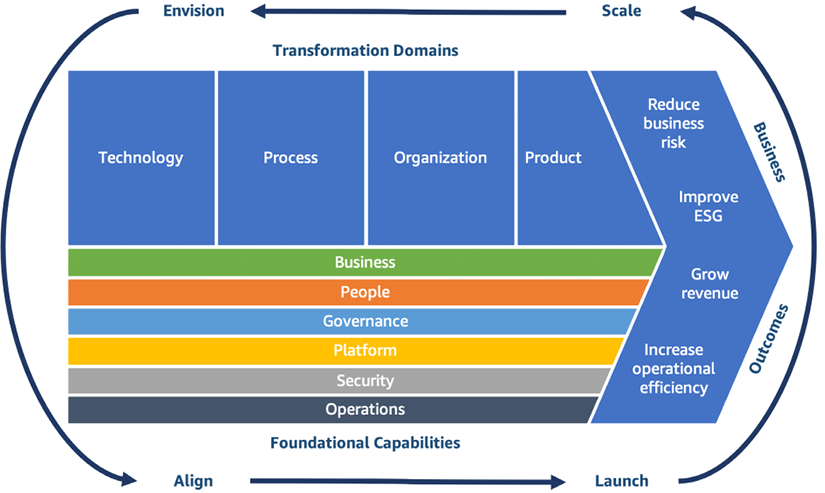
AWS provides the **Cloud Adoption Framework** (**CAF**) to help you start with digital transformation. Let’s learn more details about it.

**The AWS Cloud Adoption Framework (AWS CAF)**

As discussed in the previous section, cloud adoption has some critical pitfalls. The organization may have started well with the pilot but could not move it further, or tech leadership may need to be aligned to focus on cloud modernization. In some cases, even if an organization migrates to the cloud, it cannot realize its full value, as replicating the on-premise model to the cloud can fail to reduce cost or increase flexibility. To help customers overcome these pitfalls, AWS designed the Cloud Adoption Framework by applying their learning across thousands of customers who completed their cloud migration to AWS.

The AWS Cloud Adoption Framework is a mechanism for establishing a shared mental model for cloud transformation. It utilizes AWS’s experience and best practices to enable customers to build business transformation in the cloud. It further helps validate and improve cloud readiness while evolving your cloud adoption roadmaps.

The CAF helps to identify business outcomes such as risk, performance, revenue, and operational productivity. The following diagram provides a full view of the AWS CAF:



*Figure 3.3: AWS Cloud Adoption Framework (CAF)*

As shown in the preceding diagram, the AWS CAF proposes four incremental and iterative phases for organizations to succeed in their digital transformation journey:

* **Envision** – Understand business transformation opportunities with your strategic goals, and take buy-ins from senior executives to drive change. Define quantified business outcomes to drive value.
* **Align** – Identify gaps and dependencies across the organization to create a plan for cloud readiness and drive stakeholder alignment at all organizational levels.
* **Launch** – Build proof of concept and deliver impactful, successful pilots who can define future directions. Adjust your approach from pilot projects to build a production plan.
* **Scale** – Scale pilots to take it to production and realize continuous business value.

You can learn more about the AWS CAF by visiting their page: <https://aws.amazon.com/professional-services/CAF/>.

Further, the AWS CAF identifies four transformation domains that help customers accelerate their business outcomes. The following are the digital transformation opportunities:

* **Technology transformation** using cloud migration and modernization approach in the cloud.
* **Process transformation** using a data and analytics approach with cloud technology.
* **Organizational transformation** by building an efficient operating model in the cloud.
* **Product transformation** by building cloud-focused business and revenue models.

Transformation domains are enabled by a set of foundational capabilities that offer expert advice on optimizing digital transformation by harnessing the power of cloud services. The AWS CAF organizes these capabilities into six perspectives: business, people, governance, platform, security, and operations, with each perspective encompassing a distinct set of capabilities that are managed by various stakeholders involved in the cloud transformation process. By leveraging these capabilities, customers can improve their cloud readiness and transform their operations effectively.

Every organization’s cloud journey is unique. To succeed in their transformations, organizations must define their desired cloud transformation state, understand cloud readiness, and close the gaps. However, driving digital transformation through cloud adoption is not new, and many organizations have already implemented it. So, you don’t need to reinvent the wheel and can take advantage of the learning offered by AWS.

Now that we have examined computing models (IaaS, PaaS, and SaaS) and migration strategies, the question remains: what architecture should you use in the cloud?

**Architectures to provide high availability, reliability, and scalability**

We have come a long way in making our systems more reliable, scalable, and available. It wasn’t that long ago that we didn’t think of saving precious photographs and documents on our PC hard drives, assuming disk drives would be able to store this data indefinitely. Even though PC components have decent reliability, they will eventually fail. It’s the nature of hardware with moving parts such as disk drives.

Since then, significant advances have been made to increase the reliability of individual components; however, the real increase in reliability comes from redundantly storing information on multiple devices and in different locations. Doing so increases reliability exponentially.

For example, the S3 Standard service stores file redundantly with at least six copies and in at least three data centers. If a copy is corrupted, the S3 storage system automatically detects the failure and replicates the file using one of the remaining uncorrupted copies. Just like that, the number of copies for a file remains constant. So, for S3 to lose a file, all six replicas must fail simultaneously. The likelihood of this happening naturally is extremely rare. We will learn more about S3 and AWS’s other storage options in *Chapter 5*, *Storage in AWS – Choosing the Right Tool for the Job*, but for now it is enough to know that AWS offers many different ways to redundantly store data.

**NOTE**

The concept of copying data across resources to increase reliability and availability is known as **redundancy**. Redundancy is easy to implement with copies of files and objects. It is a much more difficult problem to implement with databases. The reason it’s hard is that replicating the state across machines is challenging.

It is also important to note that in the database context, redundancy has two meanings. One being *“bad”* redundancy and the other being *“good”* redundancy. Many of the database services that AWS offers provide *“good”* redundancy out of the box. Some of these services can easily and automatically replicate data for you. For example, Amazon DynamoDB automatically replicates data as it is inserted or updated. Another example is the Amazon RDS system’s capability to create read replicas easily. These processes are completely transparent to the user of the service and the administrators and are guaranteed to be eventually consistent.

Examples of *“bad”* redundancy are unnecessarily denormalized database tables or manual copies of files. Using these methods to create redundancy will most likely lead to inconsistent data, inaccuracies, and erroneous analysis of your data.

In the following three subsections, we will learn about four different types of application architectures:

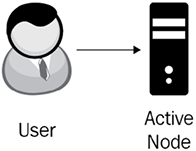
* Active architecture
* Active/passive architecture
* Active/active architecture
* Sharding architecture

Each one has different advantages and disadvantages. Let’s learn about each one of them in more detail.

**Active architecture**

In this architecture, there is only one storage resource with a single point of failure. An architecture like this can be described as functional architecture. If your hard drive fails, you are out of luck. This name might not seem entirely intuitive. You can imagine this architecture as a circus performer working without a net. If something fails, there is no backup recovery plan.

The following diagram illustrates the architecture. We have only one **Active Node**, and if any hardware failure occurs with the node, the whole system fails:



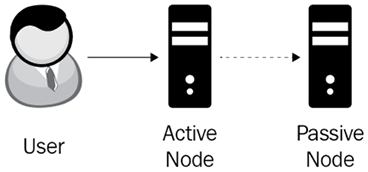
*Figure 3.4: Active architecture*

In the next section, let’s look at the active/passive architecture.

**Active/passive architecture**

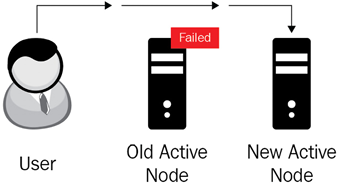
The next logical evolution to implement a more available architecture is to have a simple backup. Instead of having just one resource, a simple solution is to let a primary server (the active node) handle reads and writes and synchronize its state on a secondary server (the passive node). This is known as active/passive architecture.

As we can see in the following diagram, the system is composed of two resources. During regular operation, users communicate with the **Active Node**, and any changes to the **Active Node** get replicated to the **Passive Node**:



*Figure 3.5: Active/passive architecture*

An active/passive architecture improves availability by having a fresh copy of all your critical data. As we see in the following diagram, if the active node fails, you can manually or automatically redirect the traffic to the passive node. In this case, the passive node becomes the active node, and then you can take the necessary steps to fix or replace the failed node. There will be a period of time when you are replacing the failed node, and the whole system can fail if the new active node fails before you can replace the failed node. The following diagram illustrates this process:



*Figure 3.6: Active/passive architecture with an active down node*

The first generations of active/passive architectures used a synchronous transaction process. Transactions were not committed until the passive node acknowledged that it had processed the writes. This was not a suitable solution. If the passive node went down, it became a bottleneck in the architecture. This architecture can actually decrease the system’s reliability because now, two components can fail, bringing the whole system down.

To improve availability, later generations of this architecture used asynchronous replication.

**Asynchronous replication** is a store-and-forward method to back up data. Asynchronous replication stores data in primary storage first (synchronously). After this, it sends a request to write the data in a secondary storage location without waiting for this second write to finish. This approach speeds up the storage process. If the write to the second location fails, the process will keep trying multiple times until it succeeds. It doesn’t matter if it takes a couple of tries because the requester is not waiting for the process to finish.

While the high-level architecture looks quite similar, it can now handle failures in the active or passive nodes while continuing to process transactions.

The drawbacks of this architecture are as follows:

* The system will still fail if both nodes fail during similar timeframes.
* Any data not replicated to the passive node when the active node goes down will be lost.
* Since the passive node is used just for backup purposes, the performance and throughput of the system are limited by the capacity of the active node, and the capacity of the passive node is wasted because it is not handling any user traffic.

As applications became more complex and started handling worldwide internet traffic, and user expectations grew to have *“always on”* availability, the active/passive architecture could not scale to handle these new demands, and new architecture was needed.

The following architecture we will learn about avoids some of the shortcomings of the active/passive architecture at the expense of added complexity and additional hardware.

**Active/active architecture**

In active-active architecture, you replicate your workload to another set of nodes. When a user sends the request, you can distribute the traffic between these two nodes. If one node goes down, you always have other nodes up and running to serve user traffic. To achieve reliability, you need to ensure workload replication happens in the physically separated region, so if something happens to one data center, it should not impact others.

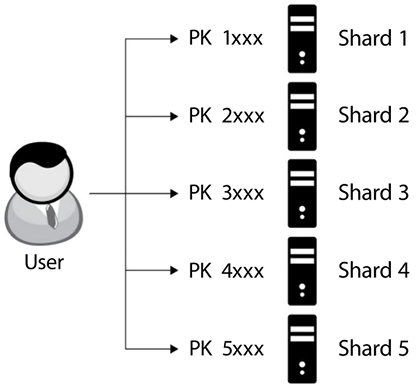
AWS provides the ability to easily configure active-active architecture across its globally distributed infrastructure and route traffic between them using its DNS service Amazon Route 53 and content distribution service AWS CloudFront. Full active-active architecture may come with a cost if you replicate the entire workload to achieve 100% fault tolerance and high availability. However, if your system doesn’t need to be fully fault-tolerant, you can use less costly options such as warm stand-by, where you can keep low-capacity servers in a secondary site and divert a small amount of traffic, like 5%. If the primary site goes down, you can quickly scale the secondary site to full capacity and make it primary.

Let’s look at another way to achieve high availability.

**Sharding architecture**

Building on the active/passive architecture, engineers developed a new architecture where there are multiple nodes, and all nodes participate in handling traffic. This architecture divides the work using a scheme to parcel out the work. One method is to divide the work using a primary key. A primary key is a number, and you have ten shards. We could set up the process so requests with a primary key starting with 1 would be sent to the first shard. If it begins with a 2, it goes to the second shard, and so on.

The following figure shows an example of sharding architecture:



*Figure 3.7: Sharding architecture*

As you may have already guessed, if you use this scheme, you should ensure that your keys are balanced and that there is a relatively even distribution of transactions across all the numbers. An example of a potentially unbalanced key would be if you have a polling application, your table stores the votes, and your key is the candidates in the election. It is possible that only a few of the candidates will get the majority of the votes.

Also, as you can imagine, **sharding architecture** increases throughput, increases availability, and reduces the number of system failures. One drawback is that a sharding system can be complex to implement. However, there are more services such as DynamoDB that implement sharding managed by AWS, so you do not need to worry about the sharding implementation details and can focus purely on the business logic of your application.

As we saw in this section and learned about these three common architectures, each has different advantages and disadvantages. Whenever possible, an active architecture should be avoided except for simple projects that are not mission-critical and for development work. When using an active architecture, we are risking the loss of data. Active architectures are inherently *“single-point-of-failure”* architectures. However, as expected, architectures that offer higher availability are more expensive solutions that require more resources.

The architectures discussed in this section are simple, but powerful, and these basics are worth keeping in the back of your mind as we discuss different ways to use AWS throughout coming chapters. However, regardless of how you choose to shape your system, it is worth making sure that it is as resilient as possible. With this in mind, let’s finish our discussion with a quick look at chaos engineering.

**Chaos engineering**

**Chaos engineering** is a methodology devoted to building resilient systems by purposely trying to break them and expose their weaknesses. It is much better to deal with a problem when we expect it to happen. A well-thought-out plan needs to be in place to manage failure that can occur in any system. This plan should allow the system’s recovery in a timely manner so that our customers and our leadership can continue to have confidence in our production systems.

One element of this plan could be having a clearly established **Recovery Point Objective** (**RPO**) and **Recovery Time Objective** (**RTO**). The concepts of RPO and RTO, in addition to an analysis of the business impact, will provide the foundation to nail down an optimal recommendation for a business continuity plan. The strategy should ensure that normal business operations resume within an acceptable time frame that meets or exceeds the RPO and RTO for the agreed-upon **Service Level Agreement** (**SLA**).

**RTO** is the targeted time that can elapse after a problem occurs with a resource and before it again becomes accessible. In other words, it is the amount of time needed to recover and make a service available to maintain business continuity. The time elapsed should not exceed that which is specified in an SLA. This time interval is called **tolerance**. For example, if your SLA specifies that the accounting database cannot be unavailable for more than 60 minutes, there is an outage, and the database is back up and running within 40 minutes, then that outage has met the RTO.

**RPO** refers to how old the data that is restored can be. In other words, it is the time between a data corruption incident and the last backup taken. This will be closely tied to the architecture used and how often data is replicated. Any data that is lost that was not saved during this period will be lost and will have to be manually re-entered. For example, say a database is automatically backed up every five minutes and a component fails causing the database to go down. After this, if it is determined that 3 minutes worth of updates are lost because they weren’t written to the backup, and the SLA specifies an RPO of 10 minutes then the 3 minutes of lost data are within an acceptable range as dictated by the SLA.

However, RTO and RPO are only objectives – meaning they is what we expect will happen. They need to be compared with the **Recovery Point Actual** (**RPA**) and **Recovery Time Actual** (**RTA**) which describe what actually happened. In other words, RTO and RPO are estimates of what will happen. RPA and RTA are measurements of what actually occurred. If the actual time (RTA) it takes to recover is longer than the objective (RTO), it means that your SLA was not met, and you need to do better. To ensure you are ready, a plan must be put in place to test your recovery steps and ensure you meet the objectives.

Establishing an RTO and RPO is one example of what could be included in your plan to manage failure. No matter what steps are taken in your plan, it is highly recommended to schedule planned outages to test it. This is where chaos engineering comes into play.

A common refrain is that “*we learn more from failure than we learn from success*”. Chaos engineering takes this refrain and applies it to computing infrastructure. However, instead of waiting for failure to occur, chaos engineering creates these failure conditions in a controlled manner to test the resiliency of our systems.

Systemic weaknesses can take many forms. Here are some examples:

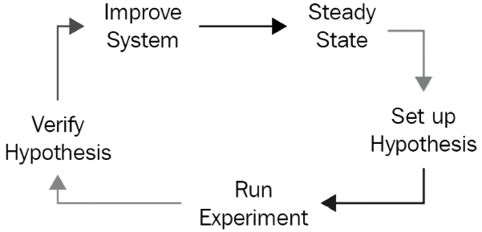
* Insufficient or non-existent fallback mechanisms any time a service fails.
* Retry storms result from an outage and timeout intervals that are not correctly tuned.
* Outages from downstream dependencies.
* A single-point-of-failure crash in upstream systems causes cascading failures.

Retry storms can occur when timeout intervals and outage handling are not appropriately configured, leading to a high volume of repeated requests to a service that is experiencing issues. When a service outage occurs, the client may attempt to retry the failed request, which can lead to an overwhelming number of requests being sent to the service when it is already experiencing high traffic or is not available. This increased traffic can further worsen the outage, leading to a vicious cycle of repeated retries and degraded service performance. Therefore, it is crucial to properly configure timeout intervals and retry policies to prevent retry storms and minimize the impact of outages.

Chaos engineering is a series of experiments to continuously test fragility and find weaknesses in our systems to harden and reinforce them. The steps that can be taken to implement each one of these experiments are the following:

1. Define the *“steady state”* of the system. This is a set of metrics for the system that signals what expected behavior should be.
2. Make an assumption (the hypothesis) that this steady state will continue to prevail under *“normal conditions”* (the control group) as well as under *“abnormal conditions”* (the experimental group).
3. Introduce *“chaos”*. Change conditions that mimic real-world events such as a server crash, a hard drive malfunction, severed network connections, system latency, etc.
4. Attempt to disprove the hypothesis by observing the differences between the steady state (the control group) and the altered state (the experimental group).
5. Make improvements to the system based on the results observed from running the experiment.

The following diagram illustrates this cycle of steps:



*Figure 3.8: Chaos engineering cycle*

If we cannot disrupt the steady state after introducing chaos (traffic still flows and users are still serviced), this will increase our confidence in the system. Whenever a weakness is found, appropriate measures can be taken to close the gap and eliminate the weakness. But it happens under controlled conditions, on our terms rather than in the form of a phone call or a text at 3 A.M. This is the perfect application of the adage. *You learn more from failure than success*.

When you introduce a form of chaos that cannot break the system, it’s proof that your system can handle that chaotic scenario. Only when an experiment makes the system fail do you realize you have a weakness or vulnerability that needs to be addressed. As such, chaos engineering is a valuable method for testing the strength of your cloud environment and improving it.

**Summary**

This chapter taught you about cloud computing models, such as IaaS, PaaS, and SaaS. You dove deep into each model to understand its value proposition and how it can fit your IT workload needs.

You then pivoted to understand how to implement a digital transformation program and migrate to the cloud. You learned that not all cloud migration programs are created equal and that some companies use their migration to the cloud to implement further change in their organization. You also learned about cloud migration strategy and how the AWS 7 Rs migration strategy could help you build a migration and modernization plan.

Since processes will have to be retested anyway, why not take advantage of this and implement ways to improve the company’s processes and business workflows?

You also covered the drivers for digital transformation and visited some examples of successful digital transformation. You saw some valuable tips to ensure the success of your digital transformation and some pitfalls that should be avoided so that your transformation runs smoothly. You further learned how the AWS **Cloud Adoption Framework** (**CAF**) could help you to accelerate a successful digital transformation journey.

Finally, you examined a few simple architectures that can be utilized in the cloud to improve availability, reliability, and scalability. These included the active architecture, active/passive architecture, active/active architecture, and sharding architecture. You then finished the chapter with a brief look at a method to improve your architecture called chaos engineering.

In the next chapter, you will learn how to use AWS global infrastructure to create business solutions that can help you with your digital transformation initiatives. You will explore technology and learn about networking in AWS, a foundational pillar of starting your AWS technical journey.