Application portfolio assessment guide for AWS Cloud migration

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Application portfolio assessment guide for AWS Cloud migration

Table of Contents

Application portfolio assessment guide for AWS Cloud migration

German Goncalves and Mark Berner, Amazon Web Services (AWS)

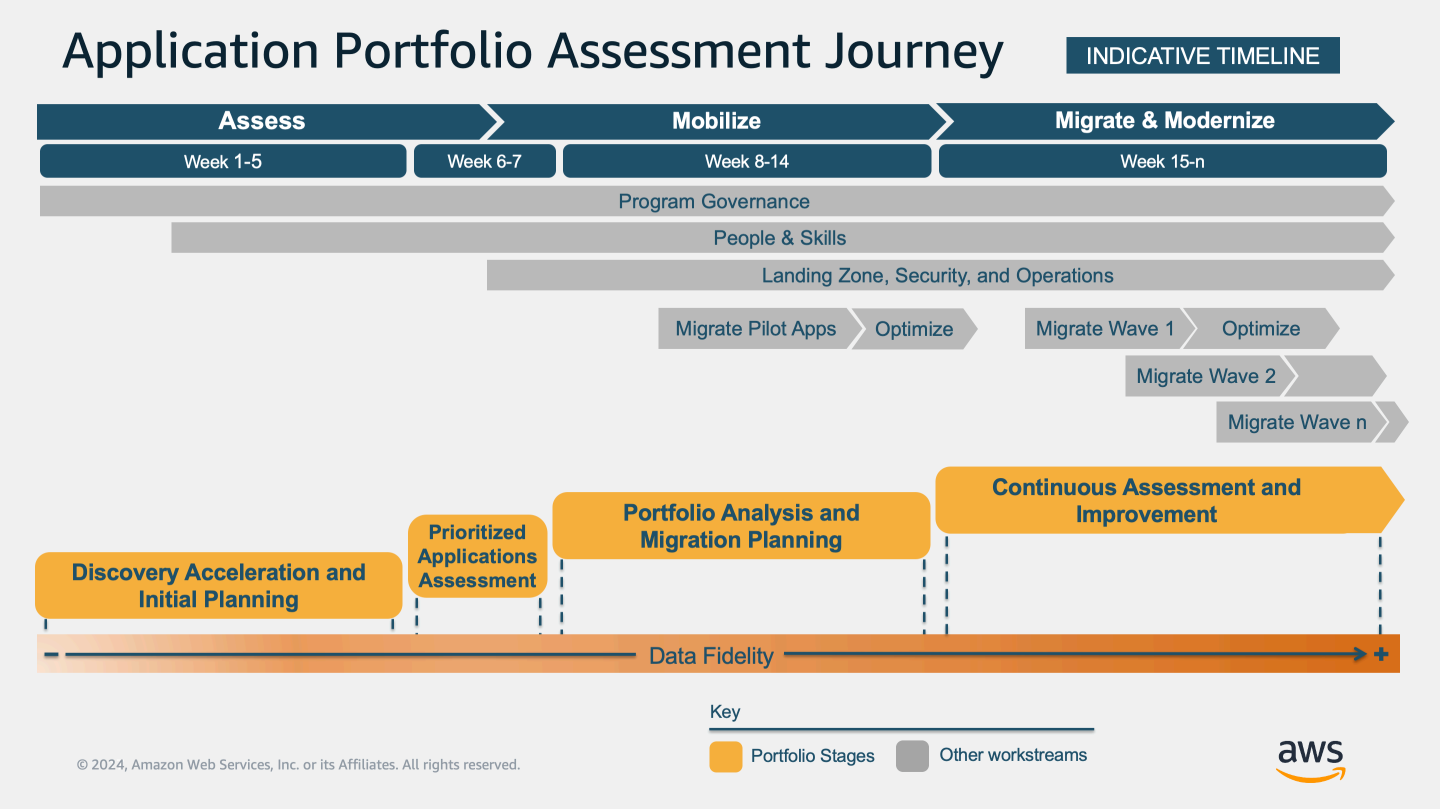
May 2024 (document history)

This Amazon Web Services (AWS) Prescriptive Guidance document dives deep into implementing the application portfolio assessment strategy. You can use this guide to help you initiate and progress through the assessment of your portfolio of applications and associated infrastructure. The assessment includes discovery, analysis, and planning. Infrastructure includes compute, storage, and networks.

Overview

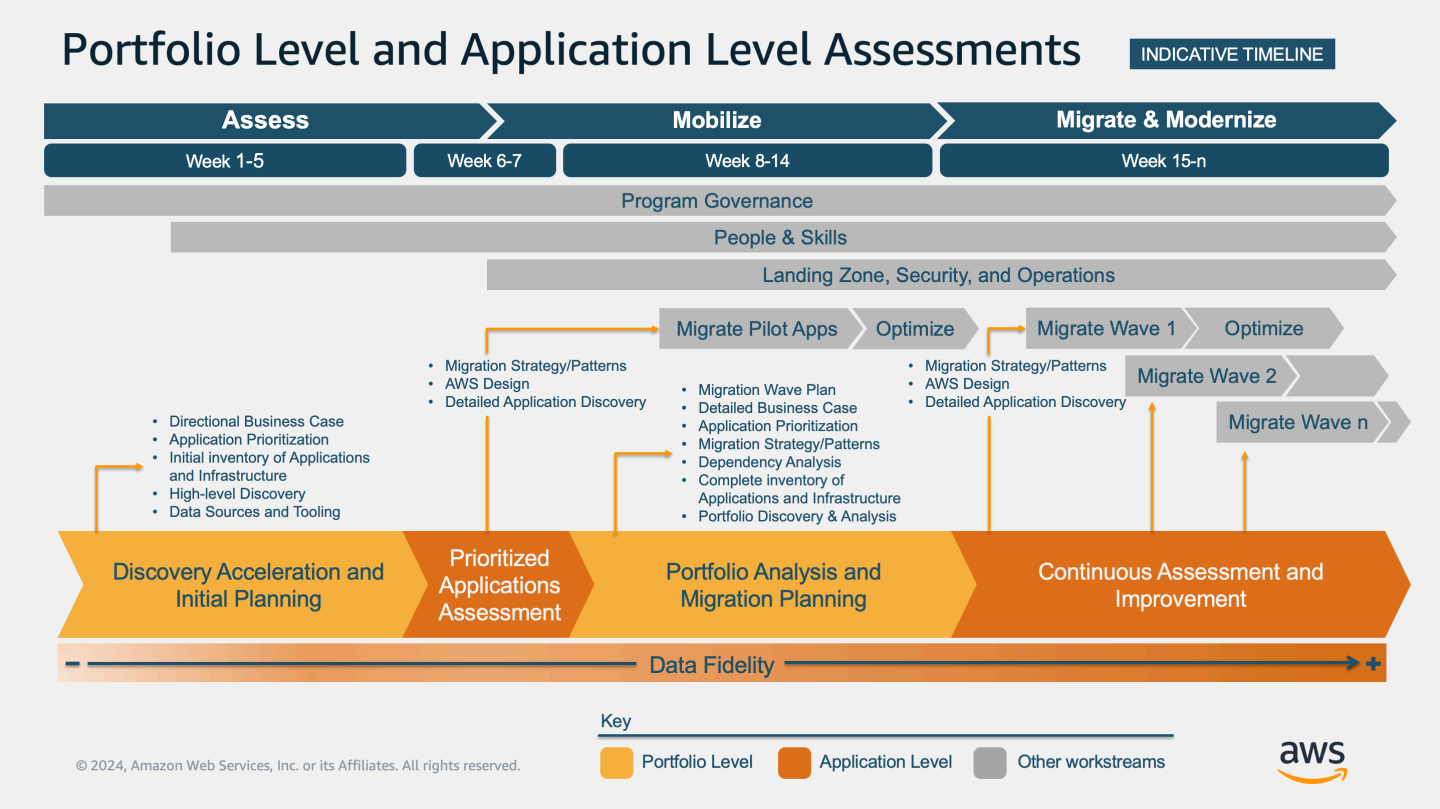
Long-running cloud migration programs require the coordination of several workstreams such as program governance, landing zone (an operative target environment with security controls), migration, and application portfolio. The names of these workstreams might vary depending on how you choose to organize your migration program. As a workstream, application portfolio assessment represents a foundational activity across the entire lifecycle of these programs. The understanding of the portfolio that is gained through the assessment provides a key input to other workstreams that depend on the data and analysis resulting from the continuous application portfolio assessment.

The following diagram shows how the stages of portfolio assessment correspond to the AWS phases of migration and other workstreams. The portfolio discovery and initial planning stage begins in the assess phase, typically during the first five weeks. Prioritized applications assessment, in the sixth and seventh weeks, spans the assess and mobilize phases. The portfolio analysis and migration planning stage happens in weeks 8-14, in the mobilize phase. The continuous assessment and improvement stage happens in the migrate and modernize phase, from week 15 until the end of the migration program. This timeline is indicative. The actual duration of the stages will depend on the overall program organization. The portfolio assessment stages are also valid outside of this framework, and they can be incorporated into any migration program structure.



- **Discovery acceleration and initial planning** focuses on the current understanding of the portfolio. It includes creating a directional business case, establishing base rationalization models for migration, and identifying initial migration candidates.
- **Prioritized applications assessment** delivers faster time-to-value through a detailed assessment, initial design of the target state architecture, and identification of applications that can be moved in the short term. Moving applications quickly provides teams with migration experience and establishes cloud foundations, such as an initial landing zone and other infrastructure components.
- **Portfolio analysis and migration planning** focuses on building a complete and up-to-date view of the application portfolio. The view is built by iteratively enriching the portfolio dataset, closing data gaps, evolving the business case, and creating high-confidence migration wave plans.
- **Continuous assessment and improvement** supports migrations at scale by producing detailed application and technology assessments for each migration wave as a continuous activity. This stage includes iterating the migration wave plan and conducting further analysis of migrated workloads for optimization and modernization.

The following diagram shows the key activities for each stage of assessment and how they pivot between portfolio-level assessment and application-level assessment. Portfolio-level assessment focuses on high-level discovery and overall analysis of the portfolio. For example, sources of portfolio data, application and infrastructure inventory, prioritization, and directional business case. Application-level assessment focuses on the detailed discovery of one or more applications. For example, detailed application discovery, target AWS design, and migration strategy at the architecture and technology levels of the applications. Portfolio-level and application-level assessments represents the breadth and depth of information required.



Discovery acceleration and initial planning

This first stage of portfolio assessment focuses on the initial steps of obtaining and analyzing data at the portfolio level. The main objective is to identify business drivers and collect general data from applications and infrastructure to obtain an initial view of the portfolio. This data includes high-level technical and business attributes such as application names, environment, product versions, criticality, performance values, and others, as described in the data requirements section. Completing this stage is key to understanding the scope of the project, identifying initial migration candidates, and informing the business case.

Primary outcomes of this stage

- Documented business drivers, outcomes, goals, and technical guiding principles.
- An initial inventory of applications and infrastructure, and identified data gaps. This is an initial view of the portfolio that will be iterated and refined in further stages.
- A directional business case and estimated cost to migrate.
- A list of initial migration candidates (for example, three-five applications).
- Defined next steps.

Understanding initial assessment data requirements

Data collection can take a significant amount of time and easily become a blocker when there is no clarity about what data is needed and when it is needed. The key is to understand the balance between what is too little and what is too much data for the outcomes of this stage. To focus on the data and the fidelity level required for this early stage of portfolio assessment, adopt an iterative approach to data collection.

Data sources and data requirements

The first step is to identify your sources of data. Start by identifying the key stakeholders within your organization that can fulfill the data requirements. These are typically members of the service management, operations, capacity planning, monitoring, and support teams, and the application owners. Establish working sessions with members of these groups. Communicate data requirements and obtain a list of tools and existing documentation that can provide the data.

To guide these conversations, use the following set of questions:

- How accurate and up to date is the current infrastructure and application inventory? For example, for the company configuration management database (CMDB), do we already know where the gaps are?
- Do we have active tools and processes that keep the CMDB (or equivalent) updated? If so, how frequently it is updated? What is the latest refresh date?
- Does the current inventory, such as the CMDB, contain application-to-infrastructure mapping? Is each infrastructure asset associated to an application? Is each application mapped to infrastructure?
- Does the inventory contain a catalog of licenses and licensing agreements for each product?
- Does the inventory contain dependency data? Note the existence of communication data such as server to server, application to application, application or server to database.
- What other tools that can provide application and infrastructure information are available in the environment? Note the existence of performance, monitoring, and management tools that can be used as a source of data.
- What are the different locations, such as data centers, hosting our applications and infrastructure?

After these questions have been answered, list your identified sources of data. Then assign a level of fidelity, or level of trust, to each of them. Data validated recently (within 30 days) from active programmatic sources, such as tools, have the highest level of fidelity. Static data is considered of lower fidelity and less trusted. Examples of static data are documents, workbooks, manually updated CMDBs, or any other non-programmatically maintained dataset, or whose last refresh date is older than 60 days.

The data fidelity levels in the following table are provided as examples. We recommend that you assess the requirements of your organization in terms of maximum tolerance to assumptions and associated risk to determine what is an appropriate level of fidelity. In the table, institutional knowledge refers to any information about applications and infrastructure that is not documented.

Data sources	Fidelity level	Portfolio coverage	Comments
Institutional knowledge	Low - Up to 25% of accurate data, 75% assumed values or	Low	Scarce, focused on critical applications

Data sources	Fidelity level	Portfolio coverage	Comments
	data is older than 150 days.		
Knowledge base	Medium-low - 35-40% of accurate data, 65-60% assumed values or data is 120-150 days old.	Medium	Manually maintained, inconsistent levels of detail
CMDB	Medium - ~50% of accurate data, ~50% assumed values or data is 90-120 days old.	Medium	Contains data from mixed sources, several data gaps
VMware vCenter exports	Medium-high - 75-80% of accurate data, 25-20% assumed values or data is 60-90 days old.	High	Covers 90% of the virtualized estate
Application performance monitoring	High - Mostly accurate data, ~5% assumed values or data is 0-60 days old.	Low	Limited to critical production systems (covers 15% of the application portfolio)

The following tables specify the required and optional data attributes for each asset class (applications, infrastructure, networks, and migration), the specific activity (inventory or business case), and the recommended data fidelity for this stage of assessment. The tables use the following abbreviations:

- R, for required
- (D), for directional business case, required for total cost of ownership (TCO) comparisons and directional business cases

- (F), for full directional business case, required for TCO comparison and directional business cases that include migration and modernization costs
- O, for optional
- N/A, for not applicable

Applications

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Unique identifier	For example, application ID. Typically available on existent CMDBs or other internal inventories and control systems. Consider creating unique IDs whenever these are not defined in your organization.	R	R (D)	High
Application name	Name by which this application is known to your organization. Include commercial off-the-shelf (COTS) vendor and product name when applicable.	R	R (D)	Medium-high

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Is COTS?	Yes or No. Whether this is a commercial application or internal development	R	R (D)	Medium-high
COTS product and version	Commercial software product name and version	R	R (D)	Medium
Description	Primary application function and context	R	O	Medium
Criticality	For example, strategic or revenue-generating application, or supporting a critical function	R	O	Medium-high
Type	For example, database, customer relationship management (CRM), web application, multimedia, IT shared service	R	O	Medium

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Environment	For example, production, pre-production, development, test, sandbox	R	R (D)	Medium-high
Compliance and regulatory	Frameworks applicable to the workload (e.g., HIPAA, SOX, PCI-DSS, ISO, SOC, FedRAMP) and regulatory requirements	R	R (D)	Medium-high
Dependencies	Upstream and downstream dependencies to internal and external applications or services. Non-technical dependencies such as operational elements (e.g., maintenance cycles)	O	O	Medium-low

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Infrastructure mapping	Mapping to physical and/or virtual assets that make up the application	O	O	Medium
License	Commodity software license type (e.g., Microsoft SQL Server Enterprise)	O	R	Medium-high
Cost	Costs for software license, software operations, and maintenance	N/A	O	Medium

Infrastructure

Attribute Name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Unique identifier	For example, server ID. Typically available on existing CMDBs or other internal inventories and control systems.	R	R	High

	Consider creating unique IDs whenever these are not defined in your organization.			
Network name	Asset name in the network (e.g., hostname)	R	O	Medium-high
DNS name (fully qualified domain name, or FQDN)	DNS name	O	O	Medium
IP address and netmask	Internal and/or public IP addresses	R	O	Medium-high
Asset type	Physical or virtual server, hypervisor, container, device, database instance, etc.	R	R	Medium-high
Product name	Commercial vendor and product name (for example, VMware ESXi, IBM Power Systems, Exadata)	R	R	Medium

Operating system	For example, REHL 8, Windows Server 2019, AIX 6.1	R	R	Medium-high
Configuration	Allocated CPU, number of cores, threads per core, total memory, storage, network cards	R	R	Medium-high
Utilization	CPU, memory, and storage peak and average. Database instance throughput.	R	O	Medium-high
License	Commodity license type (e.g., RHEL Standard)	R	R	Medium
Is shared infrastructure?	Yes or No to denote infrastructure services that provide shared services such as authentic ation provider, monitoring systems, backup services, and similar services	R	R (D)	Medium

Application mapping	Applications or application components that run in this infrastructure	O	O	Medium
Cost	Fully loaded costs for bare-metal servers, including hardware, maintenance, operations, storage (SAN, NAS, Object), operating system license, share of rackspace, and data center overheads	N/A	O	Medium-high

Networks

Attribute Name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Size of pipe (Mb/s), redundancy (Y/N)	Current WAN link specifications (e.g., 1000 Mb/s redundant)	O	R	Medium
Link utilization	Peak and average utilization, outbound	O	R	Medium

	data transfer (GB/month)			
Latency (ms)	Current latency between connected locations.	O	O	Medium
Cost	Current cost per month	N/A	O	Medium

Migration

Attribute Name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Rehost	Customer and partner effort for each workload (person-days), customer and Partner cost rates per day, tool cost, number of workloads	N/A	R (F)	Medium-high
Replatform	Customer and partner effort for each workload (person-days), customer and partner cost rates per day,	N/A	R (F)	Medium-high

	number of workloads			
Refactor	Customer and partner effort for each workload (person-days), customer and partner cost rates per day, number of workloads	N/A	O	Medium-high
Retire	Number of servers, average decommission cost	N/A	O	Medium-high
Landing zone	Re-use existing (Y/N), list of AWS Regions needed, cost	N/A	R (F)	Medium-high
People and change	Number of staff to train in cloud operations and development, cost of training per person, cost of training time per person	N/A	R (F)	Medium-high
Duration	Duration of in-scope workload migration (months)	O	R (F)	Medium-high

Parallel cost	Time frame and rate at which as-is costs can be removed during migration	N/A	O	Medium-high
	Time frame and rate at which AWS products and services, and other infrastructure costs, are introduced during migration	N/A	O	Medium-high

Evaluating the need for discovery tooling

Does your organization need discovery tooling? Portfolio assessment requires high-confidence, up-to-date data about applications and infrastructure. Initial stages of portfolio assessment can use assumptions to fill data gaps.

However, as progress is made, high-fidelity data enables the creation of successful migration plans and the correct estimation of target infrastructure to reduce cost and maximize benefits. It also reduces risk by enabling implementations that consider dependencies and avoids migration pitfalls. The primary use case for discovery tooling in cloud migration programs is to reduce risk and increase confidence levels in data through the following:

- Automated or programmatic data collection, resulting in validated, highly trusted data
- Acceleration of the rate at which data is obtained, improving project speed and reducing costs
- Increased levels of data completeness, including communication data and dependencies not typically available in CMDBs
- Obtaining insights such as automated application identification, TCO analysis, projected run rates, and optimization recommendations
- High-confidence migration wave planning

When there is uncertainty about whether systems exist in a given location, most discovery tools can scan network subnets and discover those systems that respond to ping or Simple Network Management Protocol (SNMP) requests. Note that not all network or systems configurations will allow ping or SNMP traffic. Discuss these options with your network and technical teams.

Further stages of application portfolio assessment and migration heavily rely on accurate dependency-mapping information. Dependency mapping provides an understanding of the infrastructure and configuration that will be required in AWS (such as security groups, instance types, account placement, and network routing). It also helps with grouping applications that must move at the same time (such as applications that must communicate over low latency networks). In addition, dependency mapping provides information for evolving the business case.

When deciding on a discovery tool, it is important to consider all stages of the assessment process and to anticipate data requirements. Data gaps have the potential to become blockers, so it is key to anticipate those by analyzing future data requirements and data sources. Experience in the field dictates that most stalled migration projects have a limited dataset in which the applications in scope, associated infrastructure, and their dependencies are not clearly identified. This lack of identification can lead to incorrect metrics, decisions, and delays. Obtaining up-to-date data is the first step to successful migration projects.

How to select a discovery tool?

Several discovery tools in the market provide different features and capabilities. Consider your requirements. And decide on the most appropriate option for your organization. The most common factors when deciding on a discovery tool for migrations are the following:

Security

- What is the authentication method to access the tool data repository or analytics engines?
- Who can access the data, and what are the security controls to access the tool?
- How does the tool collect data? Does it need dedicated credentials?
- What credentials and access level does the tool need to access my systems and obtain data?
- How is data transferred between the tool components?
- Does the tool support data encryption at rest and in-transit?
- Is data centralized in a single component inside or outside of my environment?
- What are the network and firewall requirements?

Ensure that security teams are involved in early conversations about discovery tooling.

Data sovereignty

- Where is the data stored and processed?
- Does the tool use a software as a service (SaaS) model?
- Does it have the possibility to retain all data within the boundaries of my environment?
- Can data be screened before it leaves the boundaries of my organization?

Consider your organization needs in terms of data residency requirements.

Architecture

- What infrastructure is required and what are the different components?
- Is more than one architecture available?
- Does the tool support installing components in air-locked security zones?

Performance

- What is the impact of data collection on my systems?

Compatibility and scope

- Does the tool support all or most of my products and versions? Review the tool documentation to verify supported platforms against the current information about your scope.
- Are most of my operating systems supported for data collection? If you don't know your operating system versions, try to narrow the list of discovery tools to those with the wider range of supported systems.

Collection methods

- Does the tool require to install an agent on each targeted system?
- Does it support agent-less deployments?
- Do agent and agent-less provide the same features?
- What is the collection process?

Features

- What are the features available?
- Can it calculate total cost of ownership (TCO) and estimated AWS Cloud run rate?
- Does it support migration planning?
- Does it measure performance?
- Can it recommend target AWS infrastructure?
- Does it perform dependency mapping?
- What level of dependency mapping does it provide?
- Does it provide API access? (for example, can it be programmatically accessed to obtain data?)

Consider tools with strong application and infrastructure dependency-mapping functions and those that can infer applications from communication patterns.

Cost

- What is the licensing model?
- How much does the licensing cost?
- Is the pricing for each server? Is it tiered pricing?
- Are there any options with limited features that can be licensed on-demand?

Discovery tools are typically used throughout the entire lifecycle of migration projects. If your budget is limited, consider at least 6 months. However, absence of discovery tooling typically leads to higher manual effort and internal costs.

Support model

- What levels of support are provided by default?
- Is any support plan available?
- What are the incident response times?

Professional services

- Does the vendor offer professional services to analyze discovery outputs?

- Can they cover the elements of this guide?
- Are there any discounts or bundles for tooling + services?

> **Tip**
>
> To find and evaluate discovery tooling, use the Discovery, Planning, and Recommendation site.

Recommended features for the discovery tool

To avoid provisioning and combining data from multiple tools over time, a discovery tool should cover the following minimum features:

- **Software** – The discovery tool should be able to identify running processes and installed software.
- **Dependency mapping** – It should be able to collect network connection information and build inbound and outbound dependency maps of the servers and running applications. Also, the discovery tool should be able to infer applications from groups of infrastructure based on communication patterns.
- **Profile and configuration discovery** – It should be able to report the infrastructure profile such as CPU family (for example, x86, PowerPC), the number of CPU cores, memory size, number of disks and size, and network interfaces.
- **Network storage discovery** – It should be able to detect and profile network shares from network-attached storage (NAS).
- **Performance** – It should be able to report peak and average utilization of CPU, memory, disk, and network.
- **Gap analysis** – It should be able to provide insights on data quantity and fidelity.
- **Network scanning** – It should be able to scan network subnets and discover unknown infrastructure assets.
- **Reporting** – It should be able to provide collection and analysis status.
- **API access** – It should be able to provide programmatic means to access collected data.

Additional features to consider

- **TCO analysis** to provide a cost comparison between current on-premises cost and projected AWS cost.
- **Licensing analysis and optimization recommendations** for Microsoft SQL Server and Oracle systems in rehost and replatform scenarios.
- **Migration strategy recommendation** (Can the discovery tool make default migration R type recommendations based on current technology?)
- **Inventory export** (to CSV or a similar format)
- **Right-sizing recommendation** (for example, can it map a recommended target AWS infrastructure?)
- **Dependency visualization** (for example, can dependency mapping be visualized in a graphical mode?)
- **Architectural view** (for example, can architectural diagrams be automatically produced?)
- **Application prioritization** (Can it assign weight or relevance to application and infrastructure attributes to create prioritization criteria for migration?)
- **Wave planning** (for example, recommended groups of applications and the ability to create migration wave plans)
- **Migration cost estimation** (estimation of effort to migrate)

Deployment considerations

After you have selected and procured a discovery tool, consider the following questions to drive conversations with the teams responsible for deploying the tool in your organization:

- Are servers or applications operated by a third party? This could dictate the teams to involve and processes to follow.
- What is the high-level process for gaining approval to deploy discovery tools?
- What is the main authentication process to access systems such as servers, containers, storage, and databases? Are server credentials local or centralized? What is the process to obtain credentials? Credentials will be required to collect data from your systems (for example, containers, virtual or physical servers, hypervisors, and databases). Obtaining credentials for the discovery tool to connect to each asset can be challenging, especially when these assets are not centralized.
- What is the network security zones outline? Are network diagrams available?
- What is the process for requesting firewall rules in the data centers?

- What are the current support service-level agreements (SLAs) in relation to data center operations (discovery tool installation, firewall requests)?

Business drivers and technical guiding principles

Business drivers

Whether your organization has already decided to move to the cloud or is close to that decision, defining and documenting business drivers for cloud migration will clarify the reasons for migrating. After the reasons are documented, you can define what will be migrated and how it will be migrated. This activity is important. We recommend that it takes place as early in the process as possible to inform and guide next steps.

Identify the stakeholders that should be part of the discussion to document the drivers. Typically, CxOs, senior managers, and key technology leaders within the organization, and your own customers. Although your customers are not likely to be part of this discussion, we recommend that one or more persons in your organization are designated represent your customers' views and goals.

Business drivers should be linked to a metric that can be measured throughout the migration journey to validate whether the outcomes have been achieved. The company's strategic goals and annual reports can act as a starting point.

Focus the conversation on where the company wants to be, based on existing and projected metrics, as a result of moving to the cloud. Consider goals and business outcomes. Also, consider what success looks like as cloud adoption increases.

Next, establish the importance level for each driver. What are the priorities? What are the expected benefits? How do the benefits support the business goals and outcomes? In the context of application portfolio assessment, the answers will help to prioritize workloads for migration and to establish technical guiding principles. However, business drivers will define and impact the migration program as a whole.

Technical guiding principles

Technical guiding principles inform migration strategy selection in later stages of portfolio assessment. In the current stage, the focus is to identify them.

Guiding principles can be established as general technology-related and approach-related decisions derived from business goals and outcomes.

For example, a company has a primary goal to reduce cost, and the desired outcome is to close an on-premises data center by a given date in 6–12 months. A resulting guiding principle is to lift and shift all applications to the cloud by using a rehost or relocate migration strategy whenever possible. In this case, the lift-and-shift approach accelerates near-term migration outcomes. After the applications have moved out of the on-premises data center, the company can focus on the main business drivers to optimize or modernize the migrated workloads.

To establish the technical guiding principles, start by analyzing business drivers. Identify a list of technologies and techniques that will achieve the business goals and outcomes. Next, refine the list and assign an order of relevance based on suitability or preference to achieve a desired outcome.

Document and communicate the guiding principles with the people involved in planning and performing the migration. Highlight concerns and potential conflicts between the principles and the actual implementation.

The following table provides an example of business drivers and technical guiding principles.

Business driver	**Outcome**	**Metrics**	**Technical guiding principle**
Accelerate innovation.	Improved competitiveness, increased business agility	Number of deployments per day or month, new features released per quarter, customer satisfaction scores, number of experiments	Refactor differentiating applications by using microservices and the DevOps operating model to increase agility and speed to market of new features.
Reduce operational and infrastructure costs.	Supply and demand matched, elastic cost base (pay for what you use)	Variation of spend over time	1. Rehost applications with infrastructure right-sizing. 2. Retire applications that have low or no utilization.

Business driver	Outcome	Metrics	Technical guiding principle
Increase operational resiliency.	Improved uptime, reduced mean time to recovery	SLAs, number of incidents	1. Replatform applications to the latest and best-supp orted operating system versions. 2. Implement high availability architect ures for critical applications.
Exit the data center.	Data-center closure by a date within 6–12 months	Speed of server migrations	Rehost applicati ons by using Cloud Migration Factory Solution.
Stay on premises, but increase agility and resiliency.	Improved competiti veness and uptime while remaining on premises	Number of deployments per day or month, new features release per quarter, SLAs, number of incidents	1. Modernize systems by extending their functionality into the cloud. 2. Assess for rehosting or replatfor ming to AWS Outposts.

Initiating data collection

Data collection is the process of gathering metadata from applications and infrastructure. The process is iterative throughout all stages of assessment. In each stage, data quantity and fidelity will increase. At this stage, the focus is on gathering general data that can help to establish an initial inventory. The inventory will be used to create a directional business case and the identification of initial migration candidates.

After the current data sources have been identified, we recommend gathering information from as many systems as possible. For more information, see the data requirements for this stage.

This approach has the benefit of helping to update the current portfolio view and the organization's knowledge of their applications and services. It also helps with determining what is targeted to move. The recommended approach is to review existing data, such as configuration management database (CMDB) outputs and information technology service management (ITSM) systems. Then construct a list of assets targeted for data collection. If your organization has complete clarity of what is in scope and out of scope for the migration, you might restrict data collection to the systems that are in scope.

When building your portfolio, consider the applications and their environments or software release lifecycles. For example, instead of identifying a customer relationship management (CRM) application and specify that it has test, dev, and prod environments, list three applications (for example, CRM-Test, CRM-Dev, CRM-Prod). Alternatively, use the CRM name but assign a unique ID to each environment and present them as separate records in your data repository. This will help with planning and tracking the migration of these environments individually. For example, you might want to migrate non-production environments first. By listing the instances of your application according to the environment, you can clearly manage and govern their transition.

During data collection, there might be uncertainty about which applications or servers are in a given data center or source location. In these cases, obtaining bare-metal and hypervisor lists from existing management tools is helpful. For example, you can connect to a hypervisor to obtain lists of virtual machines to be targeted for data collection.

Note that the initial output, when combining existing data sources, could be incomplete. The key is to perform a gap analysis in terms of data requirements for this stage and what can be obtained from existing sources. It's important to contrast percentage of completeness with level of data fidelity. Higher completeness levels from low-fidelity sources will contain several assumptions that could lead to flawed analysis. While this stage of assessment does not require the maximum data fidelity, we recommend that data sources are at least medium to medium-high fidelity. Contrast these numbers against your organization's tolerance to risk, including the use of assumptions to fill data gaps.

The gap analysis helps you understand the quantity and quality of data you are working with. The analysis also helps you to establish the level of assumptions that must be made to create a directional business case and prioritize applications for migration. Discovery tooling can help to fill the gaps and collect high-fidelity data. To increase the confidence levels in data and accelerate

migration outcomes, we recommend deploying discovery tooling as early as possible. Early action is also important because internal procurement, security, and implementation processes for new tools could require several weeks or months to complete.

We recommend establishing a communication plan or cadence and a scope-change control mechanism at this stage. This helps you to keep stakeholders informed so that they can plan ahead and mitigate risks. A key element for clear communications is to define a single source of truth for the application portfolio and associated infrastructure. Avoid keeping multiple systems of record and application and infrastructure lists. Keep data in one place (for example, a database, a tool, or a spreadsheet) that supports versioning and online collaboration, and assign an owner to it.

Prioritization and migration strategy

A key element of migration planning is to establish prioritization criteria. The point of this exercise is to understand the order in which applications will be migrated. The strategy is to take an iterative and progressive approach to evolve the prioritization model.

Prioritizing applications

This stage of assessment focuses on establishing initial criteria to prioritize low-risk and low-complexity workloads. These workloads are good candidates for pilot applications. Using low-risk, low-complexity workloads in initial migrations reduces the risk and gives teams the opportunity to gain experience. These criteria will be evolved in further assessment stages to align prioritization with business drivers when creating the migration wave plan.

The initial criteria should prioritize applications with a small number of dependencies, running in cloud-supported infrastructure, and from non-production environments. An example would be applications with 0–3 dependencies ready to rehost as-is in a development or test environment. These criteria are valid for defining the pilot applications and potentially the first and second migration waves, depending on the level of cloud adoption maturity and confidence levels.

Deciding what initial criteria to use

Select 2–10 data points to use for prioritizing your first workloads. These data points come from your initial application and infrastructure inventory (refer to the data collection section).

Next, define a score, or weight, for each possible value of each data point. For example, if the environment attribute is selected, and the possible values are production, development, and

test, each value is assigned a score, a greater number representing higher priority. Although it is optional, we recommend assigning a multiplying factor for importance or relevance to each data point. This optional step provides a higher-level differentiator to emphasize what is more important, which helps to keep the criteria aligned as you iterate on assigning scores to the values.

Based on the strategy to prioritize low-risk, simple applications for the first few migration waves, the following table shows example attributes selection and their value assignments.

Attribute (data point)	**Possible values**	**Score (0-99)**	**Importance or relevance multiplying factor**
Environment	Test	60	High (1x)
	Development	40	
	Production	20	
Business criticality	Low	60	High (1x)
	Medium	40	
	High	20	
Regulatory or compliance framework	None	60	High (1x)
	FedRAMP	10	
Operating system support	Cloud ready	60	Medium-high (0.8x)
	Unsupported in cloud	10	
Number of compute instances	1-3	60	Medium-high (0.8x)
	4-10	40	
	11 or more	20	
Migration strategy	Rehost	70	Medium (0.6x)
	Replatform	30	

Attribute (data point)	Possible values	Score (0-99)	Importance or relevance multiplying factor
	Refactor, or re-architect	10	

Make sure that you select attributes that can act as key differentiators between applications. Otherwise, the criteria will result in many workloads sharing the same priority. After you apply the model, we recommend looking at the top and bottom of the resulting ranking to see if you agree. If you don't generally agree, you can revisit the criteria that you used to score the workloads.

After you obtain a ranking, look at the distribution of scores across the entire portfolio. The scores themselves do not matter. It is the difference between scores that matters. For example, you might find that the top total score is 8,000 and the bottom score is 800. Consider plotting the resulting scores as a histogram, so you can verify that you have a good distribution. The ideal distribution looks like a standard bell curve, with a few very high-priority workloads and a few very low-priority workloads. The majority of applications will be somewhere in the middle.

Another key aspect of initial prioritization is to include internal teams or business units that show interest in being early adopters of the cloud. These could be a considerable lever in obtaining business support to migrate a given application, especially in the early days. If this is the case in your organization, include the business unit attribute in the preceding table. Assign a high score to those business units that are willing to come forward with their applications. Using the business unit attribute will help bring those applications to the top of the list.

After you agree with the resulting ranking, select the top 5–10 applications. These will be your initial application migration candidates. Refine the list so that you confirm 3–5 applications. This helps you to take a targeted approach when performing a detailed application assessment. For more information, see Prioritized applications assessment.

Determining the R type for migration

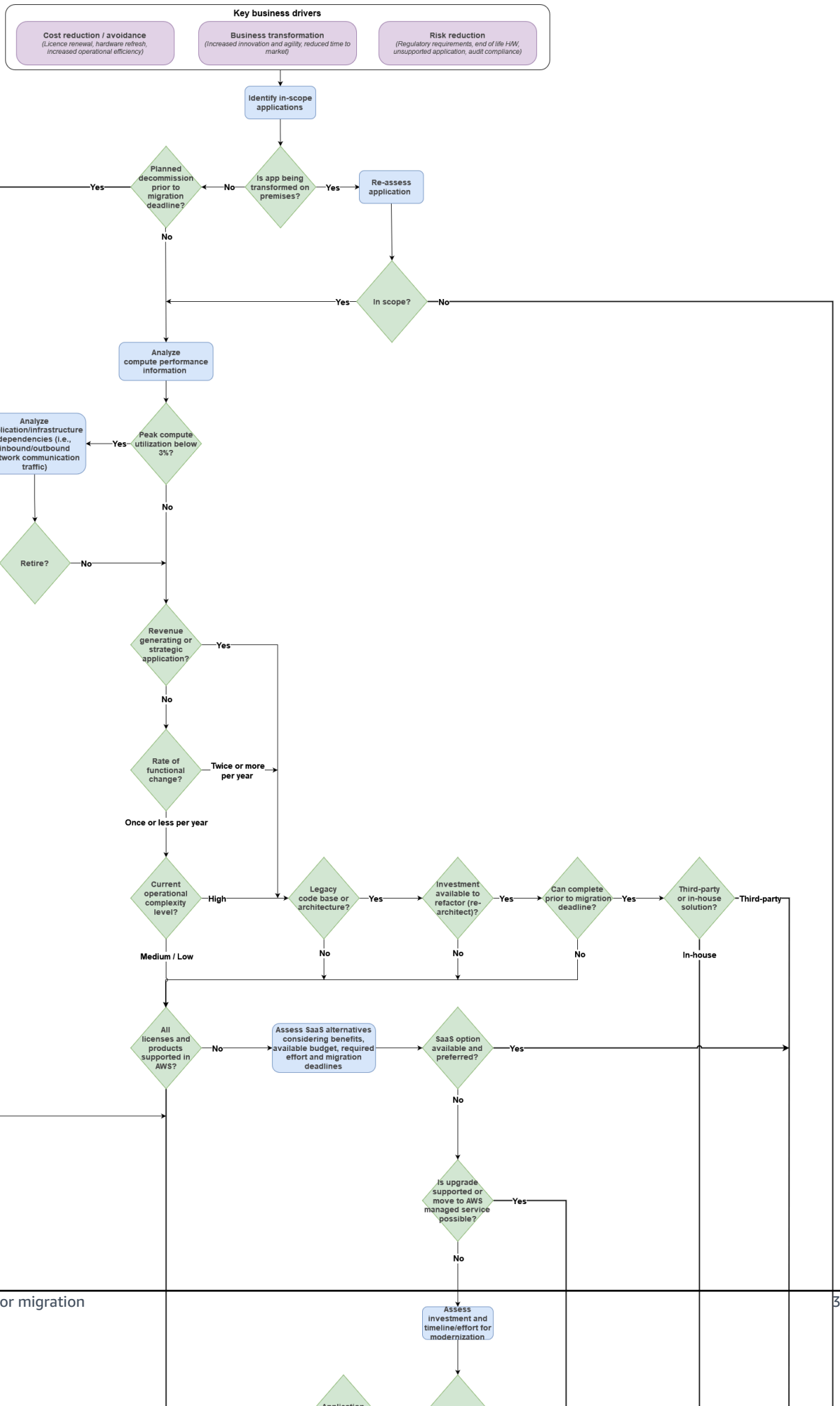
Deciding on a migration strategy for each application and associated infrastructure will have implications to migration speed, cost, and level of benefits. It is key to determine strategy based on a balanced combination of factors, including business drivers, technical guiding principles, prioritization criteria, and business strategy.

Sometimes these factors create conflicting views. For example, the primary driver for migration might be innovation and agility. At the same time, you might need to reduce costs quickly. Modernizing all applications in-scope will reduce costs in the long-run, but it will require a greater investment upfront. In that case, one approach is to migrate applications by using strategies that require less effort, such as rehost or replatform. That can provide quick efficiencies and cost reduction in the short term. Then reinvest the savings into modernizing the application at a later stage, and achieve further cost reduction.

However, starting with a complete rehost of all applications delays the greater benefits of modernization. The key is to find balance between migration strategies so that business-strategic applications are prioritized for modernization while other applications can be rehosted or replatformed first then modernized.

How to determine a migration strategy for your applications?

At this stage of assessment, the focus is to incorporate an initial model for guiding migration strategy selection. To validate the migration strategy for the initial applications, use the model in conjunction with the business drivers and the prioritization criteria. The default logic of the decision tree will help you to determine the initial treatment for the scope. In the tree, the most complex approaches, such as refactor, or re-architect, are reserved for your strategic workloads.


Key business drivers
Cost reduction / avoidance
(Licence renewal, hardware refresh, increased operational efficiency)
Business transformation
(Increased innovation and agility, reduced time to market)
Risk reduction
(Regulatory requirements, end of life H/W, unsupported application, audit compliance)
Identify in-scope applications
Is app being transformed on premises?
Yes
Re-assess application
No
Planned decommission prior to migration deadline?
Yes
No
In scope?
Yes
No
Analyze compute performance information
Peak compute utilization below 3%?
Yes
Analyze application/infrastructure dependencies (i.e., inbound/outbound network communication traffic)
No
Retire?
Yes
No
Revenue generating or strategic application?
Yes
No
Rate of functional change?
Twice or more per year
Once or less per year
Current operational complexity level?
High
Medium / Low
Legacy code base or architecture?
Yes
No
Investment available to refactor (re-architect)?
Yes
No
Can complete prior to migration deadline?
Yes
No
Third-party or in-house solution?
Third-party
In-house
All licenses and products supported in AWS?
No
Assess SaaS alternatives considering benefits, available budget, required effort and migration deadlines
SaaS option available and preferred?
Yes
No
Is upgrade supported or move to AWS managed service possible?
Yes
No
Assess investment and timeline/effort for modernization

A customizable draw.io version of this diagram is available in the *Attachments* section.

The first step to an initial model is to update the business drivers at the top of the tree with those defined by your organization. Next, apply the tree to application components rather than applications as a whole. For example, in the case of a three-tier application that has three components (front-end, application layer, and database), each component should transit the tree independently and be assigned a specific strategy and pattern. This is because in some cases you might want to rehost or replatform a given tier and refactor (re-architect) other tiers.

The independent component assignment will lead you to define a migration strategy for the associated infrastructure. The infrastructure strategy might be the same strategy as the application component that it supports, or it might be different. For example, an application component that will be replatformed into a new virtual machine with a newer operating system will follow the replatform strategy while the current virtual machine that hosts it will be retired. The migration strategy for infrastructure is calculated based on the strategy chosen for the application components.

Before using the decision tree to establish migration strategies, test the logic with a few applications and see if you generally agree with the outcome. The 6 Rs decision tree is a guide that does not replace the analysis required to determine its correctness. The tree logic might not apply to particular cases. Treat those cases as exceptions and proceed to override the decision driven by the tree by documenting the rationale for the override rather than changing the tree logic. This prevents multiple decision tree versions, which could become difficult to manage. General guidance is that the tree should be valid for at least 70-80 percent of the workloads. For the rest, there will be exceptions. Any adjustments to the tree logic, at this stage of assessment, should be focused on establishing an initial model. Further iterations and refinement will occur in later stages, such as portfolio analysis and migration planning.

Attachments

attachment.zip

Creating a directional business case

Stakeholders from across the business should understand and buy into the business case for transformation each step along the way.

In the early stages, it's important to quickly show enough potential value from a migration program, so that you can secure the resources needed to plan and establish the program. The

directional business case is designed to provide reasonable confidence in achieving compelling business value with the limited data that can be collected early.

After the program is established, the business case is developed further. The detailed case provides greater accuracy, a more complete picture of the program value, and insight into planning priorities. It defines and quantifies the planned business outcomes that the organization buys into, and it sets the baseline against which your program governance office can then steer the program and measure its achievements.

Fixing the scope of the directional business case

A directional business case is typically assembled rapidly, within 2–4 weeks. It needs to generate enough confidence so that you can secure the resources to establish the core team, engage AWS Partners if needed, and as a minimum, complete the prioritized applications assessment and portfolio analysis and migration planning stages.

Typically, directional business cases that support portfolio migrations are created as one of the following:

- A simple *total cost of ownership (TCO)* comparison between the as-is infrastructure landscape and the post-migration AWS service architecture. The comparison shows the difference in expected run rates for given workload volumes.
- A business case that shows the net present value (NPV), return on investment (ROI), payback period, modified internal rate of return (MIRR) and 3–5 year cash-flow analyses for migrating to AWS inclusive of migration costs vs staying as-is.

The directional business case scope is typically limited to one of the following:

- A comparison of the infrastructure technology costs
- A comparison of the infrastructure technology and operations costs

In general, the larger the portfolio, the less developed the case needs to be. This is because broader assumptions can be made without significantly affecting the result. For a smaller portfolio, any change will have a greater impact, so more detail is required.

Start by building the base infrastructure cost comparison. Then decide if the comparison is sufficiently compelling before you continue. Usually, portfolios of more than 400 servers will show

a positive business case on infrastructure cost reduction alone within 3 years of operation on AWS, or 250 servers within 5 years, although this can vary. For smaller portfolios, more detail may be needed.

Conversely, it is rarely useful to examine other business value components at this stage, such as value derived from improved resilience or business agility, unless the total migration scope is fewer than about 5 workloads or 50 servers.

Focus value drivers

The infrastructure technology TCO comparison compares a model of the as-is infrastructure costs with a basic model of the AWS service bill of materials needed to run your workloads with equivalent performance and availability. Many optimizations can be done. At this stage, however, the focus is on the following list because they are easier to assess and typically yield about 30 percent TCO savings, which is enough to move forward:

- **Compute elasticity** – Map servers whose usage is not 100 percent, such as development or UAT servers running 8x5 (24 percent usage), 10x5 (30 percent), or 10x6 (36 percent), and disaster recovery (DR) servers running at 2 percent, to on-demand services that are billed only when used.
- **Procure with a savings plan** – Plan to procure production servers and other servers with high usage (greater than36 percent) with a suitable savings plan to reduce costs by up to 75 percent. Options include 1-year and 3-year commitments, with differing levels of upfront payments to secure greater discounts.
- **Remove zombies** – Identify servers with CPU utilization of less than 2 percent that you can confirm are no longer needed, and remove them from the cost analysis.
- **Compute right-sizing** – Use CPU and memory utilization time series data to assess for each server the compute power and memory needed. Then select the Amazon Elastic Compute Cloud (Amazon EC2) instance to fit.
- **Relational database management system (RDBMS) license right-sizing** – Reassess your RDBMS licensing needs after compute right-sizing on your database servers, compare Bring Your Own License (BYOL) and Procuring license from AWS, and explore the potential of Amazon Relational Database Service (Amazon RDS) to increase savings.
- **Storage** – Right-size the total storage volume needed, and identify the input/output operations per second (IOPS) needs across the portfolio. Determine how much can be moved to object storage with different SLAs and costs.

Data needs

The table in Understanding initial assessment data requirements shows the data required to build each part of a directional business case, and whether it is mandatory or optional.

To build the case, you need the infrastructure subset of the initial planning data plus cost data. Determining how to identify the infrastructure to include depends on your business goal:

- If the program's objective is to migrate and modernize specific applications, build the infrastructure portfolio based on what the applications need, taking into consideration infrastructure that is shared.
- If the program's objective is infrastructure-centric, such as migrating out of a data center whose lease is due to expire, application mapping isn't needed for infrastructure TCO comparisons.

Data that is marked as optional (such as CPU and memory peak utilization for servers) can usually be replaced with standard benchmark values. You can discuss this with an AWS Partner or AWS Professional Services. Or you can extrapolate the values from data points that are available in part of your portfolio (such as data collected by a hypervisor). The larger the portfolio, the more accurate this is.

Building infrastructure TCO comparisons

Tools are vital to constructing infrastructure TCO comparisons. AWS Professional Services or an AWS Partner can provide help with all types of directional case, especially if you plan to engage them to assist in the wider migration process.

There are tools available to do the following:

- Collect inventory data.
- Collect utilization data.
- Provide accurate as-is infrastructure cost benchmarking data.
- Identify and remove zombies.
- Make right-sizing assessments.
- Recommend purchasing options.
- Compare software licensing options.
- Produce simple graphical cash-flow analyses.

Migration Evaluator from AWS is one option. It provides all of these capabilities as a **free managed service. You** can request Migration Evaluator through your AWS account manager or AWS Migration Competency Partner or by submitting a request online. Migration Evaluator has been designed specifically as a point solution to produce infrastructure technology TCO comparisons and quickly.

Key advantages:

- Free of charge
- Agent-less discovery or manual configuration of inventory data where tool-based discovery is restricted
- Dedicated support to assist deployment, configuration, data collection, and building the base case, or directional business case
- Convenience of SaaS operation, but can run data collection entirely within the customer network to support scrubbing before loading into the analytics engine
- Strong support for Microsoft license right-sizing
- Full data-export capabilities

Key limitations:

- Assesses only x86 architecture servers (Windows and Linux)
- Limited options to configure or calibrate benchmark as-is cost data
- No support for modeling operations cost optimization
- No support for migration cost modeling
- No direct support for building business cases beyond TCO comparisons

If you decide to use a commercial discovery tool for portfolio discovery and analysis capabilities such as application stack and interdependency discovery, it will usually provide infrastructure TCO comparison as well. For guidance on the use of tools for portfolio discovery and assessment, see Evaluating the need for discovery tooling. To review and compare the key capabilities of the market-leading tools, see Discovery, Planning, and Recommendation migration tools.

Building in operational cost optimization

IT operations productivity improvement is often a significant value contributor for migrations. On average, after migration to AWS, IT operational staff productivity increases by 62 percent through

migration, according to the International Data Corporation (IDC) whitepaper Fostering Business and Organizational Transformation to Generate Business Value with Amazon Web Services. However, there are two challenges with sizing and including these benefits in the directional case.

First, assessing the full range of productivity gains requires extensive data gathering and is more appropriate for the detailed business case. This challenge can be resolved by focusing on a few elements that are more easily assessed and sized with simple benchmark data but that still show significant advantage.

Second, focusing on productivity as a source of cost reduction can generate concern and negativity among key customer stakeholders and program members. Make sure that you provide clarity over how the benefit will be realized and what that means for the people impacted. Such problems can be avoided by clarifying that this will only enhance the team's roles:

- The migration program includes a track to develop and move internal operations staff into new roles, such as joining DevSecOps teams building infrastructure as code automations and test automations that will drive growth for the team.
- The benefit can be realized by rescoping and resizing operations outsourcing contracts, so that internal staff can increase their focus on higher-value activities

Approach constructing this business case element based on what operations transformations you want to consider:

- If you have an existing in-house operations team, upskill the team members, and show the expected productivity improvement.
- Alternatively, migrate away from your current operations solution to AWS Managed Services (AMS) or to an alternative managed services offering from an AWS Partner.

For the first transformation, to get a conservative financial estimate of the improved productivity that can be included in the case, we recommend the following:

1. Focus on server management operations productivity specifically. It tends to be a significant proportion of operations effort, can be more easily assessed, and is more readily verified later.
2. Calculate staffing needed based on benchmarks for number of servers that can be managed by each full-time equivalent (FTE) employee. On premises, that number is about 150 severs. On AWS, it's about 400 servers.

3. Apply these metrics to the number of on-premises servers compared with the number of EC2 instances.
4. Multiply the time saved with a blended cost rate for the whole operations team.

You can then check your results with either approach by verifying the result does not greatly exceed the average productivity gains by role provided in the following table (data sourced from the IDC whitepaper Fostering Business and Organizational Transformation to Generate Business Value with Amazon Web Services.

Role	Efficiency gain
IT infrastructure management	62%
IT support	59%
Application management	43%
Database management	19%
Application development	25%

For the second transformation, you can add the operational cost savings by directly comparing the current total operations and support cost for the in-scope portfolio with the cost of the managed service being considered.

To obtain the cost of the managed service, provide your AWS account manager or any AWS Managed Services Partner with your proposed AWS Bill of Materials, your service level choice (Plus or Premium), and your AMS package (AMS Accelerate or AMS Advanced). This will provide you with a total cost of managed services for the :AWS service components of the transformed solution. Similarly, you could obtain pricing from an AWS Partner that offers its own managed services package based its own parameters.

Expanding to a full directional business case

In general, to assemble a full directional business case, build the TCO comparison, with or without the IT productivity element, and estimate all the migration and modernization costs. Then create a cash flow that covers pairs of migrate-and-modernize and don't-migrate-and-modernize scenarios.

The most basic case is preparation of a single pair of scenarios, where the don't-migrate-and-modernize scenario is your current situation and the migrate-and-modernize scenario has the following characteristics:

- No growth or shrinkage in transactional volume, compute, or networking capacity
- Steady low-volume growth in storage requirements
- Quality-of-service capabilities (such as availability, durability, throughput, and performance) matching the existing system's capabilities

For all but very small portfolios, this fits the objectives of building a directional case well. It demonstrates enough value quickly to gain the mandate to move forward.

For smaller portfolios, it can be valuable to add pairs of migrate-and-modernize and don't-migrate-and-modernize scenarios that demonstrate other aspects of the increased value of cloud migration, such as:

- A mix of moderate and high-capacity growth requirements across workloads where that growth is expected
- Inclusion of enhanced resilience, such as high availability, DR, and fault tolerance
- Improved global performance with edge computing, content delivery network (CDN), multi-Region database replication.
- Any other specific improved quality of service that you have made a business priority for the program

For these scenarios, make sure that the costs and cash-flow implications of upgrading the current non-cloud infrastructure architecture to match the new specification are estimated accurately. The most direct way to obtain this estimate can be requesting a quotation from a systems integrator, especially if they are also an AWS Consulting Partner with Migration Competency, who can support you with both the migrate-and-modernize and the don't-migrate-and-modernize scenarios.

For each pair of scenarios, assemble a case comprising the following:

- The costs of the don't-migrate-and-modernize scenario. In the most basic case, this includes:
  - The total cost of ownership over the business case term for the current infrastructure configuration
  - Periodic increases in compute, storage, and network traffic consumption

- The costs of the migrate-and-modernize; scenario, including:
  - Setting up the program, which includes detailed discovery, migration planning, detailed business case development, establishing the core team and upskilling them, establish a landing zone if not already in place, and establishing security management and operations integration for migrated workloads
  - The workload migration and modernization costs
  - The migration infrastructure costs, including network connections, data migration services such as AWS Snowball Edge and AWS DataSync, and the AWS utility costs for the architecture needed during the migration process itself (for example, for testing)
  - The ramp-up of AWS utility costs over the course of the migration as waves go live, and the ramp down of the existing infrastructure costs as it is replaced by AWS based services and decommissioned
- The decommissioning costs and write-offs for any stranded assets

Estimating migration and modernization program setup

To set up a program for success, you might need to run a series of foundational activities to build baseline capabilities and the detailed plan if this has not been done before. These foundational activities include the following:

1. Performing detailed portfolio discovery, migration planning, and detailed business case development, as described in the Portfolio analysis and migration planning section, plus the documenting the cost of any discovery tools used.
2. Establishing a cloud business and technical core team and developing in-house skills through training and hiring. Identify the members of the IT organization that will need training, and allocate a training budget for each person.
3. Establishing a landing zone and configuring it to support the cost, operational, and security governance capabilities you will need.

AWS Consulting Partners can help provide estimates for items 1 and 3.

Estimating migration and modernization costs

To meet the objectives for a directional business case and demonstrate *just enough* commercial potential to proceed to the next phase, keep migration and modernization cost estimation as basic as possible.

To this end, we recommend that you prepare the directional business case by focusing on the applications falling into the following migration strategies:

- Retire
- Retain
- Relocate
- Rehost
- Replatform
- Repurchase

Typically, about 70 percent of workloads can be rehosted, relocated or replatformed, and another 5 percent can be retired. Assessing the applications by migration strategy usually addresses the core of the cost reduction case.

Estimating costs for refactoring, or re-architecting**,** can be complex. It is not practical to attempt this within the time frame given to preparing a directional business case. As discussed previously in Determining the R type for the migration, consider using rehost, relocate, or replatform strategies for your first phase of migration and modernization. These R strategies will likely accelerate initial payback, reduce implementation risk, and improve the business case in the short term. It is also materially easier for your application teams to modernize applications that are running within the AWS environment than those that are not. Estimates for refactoring (re-architecting) specific applications are best added when the detailed business case is prepared.

Estimating effort for migration by strategy

Each migration is different. Before committing any budgets or plans, seed workload estimates for the migration activities from the team who will be responsible for the project, whether that is your in-house applications teams, AWS Professional Services or an AWS Partner organization.

To help build the directional case, the following table provides indicative ranges of effort for the different treatments. These ranges assume that a medium-to-large portfolio is being migrated and that the migration team is trained and experienced. For small portfolios, it is best to have the team responsible for the migration prepare the estimate even for a directional case.

Migration strategy	Estimation process	Elements	Person hours	Person hours
Retain	Do nothing, with no cost, no benefits, and no reduction in technology debt.	–	–	–
Retire	Estimate decommiss ioning the hardware equipment used, if any.	–	–	–
Relocate	Estimate copying the workload within VMware using VMware tools. This includes copying the data, smoke testing to verify, and any hardware decommiss ioning. The effort to relocate VMs is typically less than for low-complexity rehost patterns.	–	–	–
Rehost	Estimate copying the	Effort per app per server	Migration	HA/DR test

Migration strategy	Estimation process	Elements	Person hours	Person hours
	workload and data with an image copy, smoke testing, high availability (HA) and disaster recovery (DR) testing where appropriate for productio n servers, and any hardware decommiss ioning. The best practice is to use tools such as AWS Applicati on Migration Service. Divide workloads into low, medium, and high complexity, based on factors such as whether a database or other infrastru cture software is running, database complexit y, whether clustered, integration	Low	10–14	3–5
		Medium	16–24	4–6
		High	26–38	8–12

Migration strategy	Estimation process	Elements	Person hours	Person hours
	complexity, and data volumes.			

Migration strategy	Estimation process	Elements	Person hours	Person hours
Replatform	For replatform migrations that include upgrades to operating system or RDBMS version, take the estimate for a rehost, and add time to run a rebuild and smoke test on the new platform.If the replatform includes changing the technology of the platform, estimate additional time for the use of the conversion tools, such as AWS Schema Conversion Tool and AWS Database Migration Service, and a more complete application test. An example of	Effort per app per server	Version up	Technology change
		Low	Add 1–3	Add 10–15
		Medium	Add 2–5	Add 20–30
		High	Add 4–8	Add 40–60

Migration strategy	Estimation process	Elements	Person hours	Person hours
	changing the technology is migrating away from a proprietary commercial database to an open source replacement.			
Repurchase	Estimate data extraction, transformation, and uploading into the newly purchased SaaS service replacement, and any hardware decommissioning.	–	–	–

Estimating migration infrastructure costs

Include estimates for the infrastructure that you will use over the course of the migration. Typically, these estimates comprise:

- A budget for connectivity and data exchange services for workload and data migration from the current environment to AWS
- A budget for the AWS services (especially compute and storage) needed for hosting the migrated workloads during the migration, testing, and cutover processes
- The ramping up of AWS utility costs as each migration wave is completed
- The decommissioning costs of the existing infrastructure that will no longer run the migrated workloads

For data exchange, examine your total data volumes and assess the feasibility of using networking. If you have provisioned an AWS Direct Connect link or AWS VPN from AWS to a point on your WAN ahead of time for operational use after the migration, you can use that resource up to its service quota.

If your network capacity is insufficient, a short-term increase in internet bandwidth with a virtual private network (VPN) is often a highly cost-effective solution. If not, AWS media exchange devices such as AWS Snowball Edge and AWS Snowball Edge offer solutions in most AWS Regions. Also, for very high-volume data migration, consider including budget for AWS DataSync, which improves reliability and can accelerate transfers irrespective of the media used.

Modeling the ramp up of AWS services and the ramp down of existing infrastructure is important for the cash flow analysis element of the business case. At this stage, you are not likely to have a wave plan to determine exactly when costs will be incurred. We recommend the following:

- Ramping up the costs for AWS at a constant rate over the migration.
- Ramping down costs for the existing infrastructure you plan to decommission at a constant rate over the same duration.

Starting the AWS cost ramp up 1-2 months before the existing infrastructure ramp down. This provides 1 month of AWS utility usage to conduct the migration for each wave. It includes time for testing, and additional time to complete the decommissioning work needed to stop incurring costs on the replaced infrastructure.

Estimating decommissioning costs

Decommissioning equipment that cannot be redeployed, and disposing of it in a legal and environmentally friendly way, can incur some small costs. However, for a directional business case, typically the only potentially material sum is the cost of writing off any remaining book value of the replaced assets.

For the directional business case, we recommend that you do the following:

- Review your asset list.
- Identify those that would be decommissioned.
- To reduce the write-off, examine the opportunities for switching devices around so that newer devices on the list can be used to replace older, more fully depreciated assets.

- Make an assessment of the future book value of the assets that would be decommissioned at that point.
- Include this as the migration cost of decommissioning.

Assembling and adjusting the full directional business case

After you prepare the full set of costs for each pair of scenarios, construct a discounted cash flow statement for each and graph them. We recommend building directional business cases over the same period as the hardware refresh cycle. This is typically 5 years for servers, storage, and network devices. When you use the same period as the hardware refresh cycle, the costs of exactly one refresh are included in the as-is costs for each scenario.

Then calculate the key financial metrics that you need for getting approval to move to the next phase of the program. We usually include the following:

- The net present value (NPV) to gauge the absolute value of the cost reductions and productivity gains assessed
- The payback period in months to verify that returns are sufficiently fast
- The final run-rate comparison to verify whether the process is taking enough cost out over the term
- The return on investment (ROI) and modified investment rate of return (MIRR) to assess the relative financial performance of the program over other demands on capital your organization may be prioritizing

Use the first iteration of the case to determine whether the expected financial performance means that refinements should be made, as in the following examples:

- If payback is too slow, consider options to accelerate and reduce the cost of migration, such as the following:
  - Use AWS Partners or AWS Professional Services to expand the available resources and further parallelize migrating workloads with more basic patterns.
  - For workloads running in VMware, compare the relocate strategy to the rehost or replatform strategy, at least for the initial phase. Using the relocate strategy can reduce migration cost and increase migration speed.
  - Where technically feasible, push workloads that require more complex replatform or refactor (re-architect) strategies into a future phase, outside the scope of the initial business case.

- If ROI and MIRR are too low, consider the following:
  - Are the scenarios that you are considering too conservative? Do you have a scenario that reflects the most likely capacity growth and elasticity needs? Do you have scenarios that compare the costs inclusive of the increases in quality of service within your objectives?
  - Can you refine the scope of the application portfolio to be migrated in the first phase to focus on workloads that will yield stronger returns, such as those with lower current utilization or expensive disaster recovery (DR) needs?
  - Can refine the scope of the application portfolio to initially exclude specific workloads that achieve less commercially? For example, can you postpone workloads for which third-party software licenses become more expensive due to different terms for deployment in public cloud infrastructure?
- If the final run-rate comparison does not meet the expected target, explore the following:
  - First, confirm that the other metrics meet expectations. The directional business case is primarily to show that there is sufficient financial opportunity to justify starting the next phase of migration preparation.
  - Identify a list of the opportunities to continue to improve cost performance on AWS after the initial phase of migration.

Include an assessment of the list of opportunities when preparing the detailed business case. In addition, include an opportunities assessment in the ongoing maintenance of the case and the month-to-month cost-optimization process after migration is complete.

Prioritized applications assessment

One of the key outcomes of the previous stage, portfolio discovery and initial planning, was to prioritize a subset of applications for detailed assessment. This section explores the detailed assessment of applications.

Looking at the details of a few applications early on will drive acceleration. The process of assessment and to-be architecture design surfaces potential blockers and clarifies important tasks that precursors to the larger-scope migration. These tasks include gathering requirements to establish AWS foundations, such as the landing zone on AWS, or to extend and validate the existing landing zone. This assessment is also the time to consider the steps and the strategy for migration.

The primary outcomes of this stage are the following:

- Validated list of prioritized applications
- Documented current state architecture
- Documented initial target architecture and migration strategy for migration candidates
- Identified migration patterns and tooling
- Documented platform requirements (security, AWS infrastructure, and operations)
- Documented cutover considerations for migration planning
- Estimated AWS run rate

Understanding detailed assessment data requirements

The following table describes the information required to obtain a complete portfolio view of the applications in the migration and their associated infrastructure.

The tables use the following abbreviations:

- R, for required
- O, for optional
- N/A, for not applicable

Applications

Attribute name	Description	Discovery, design, and migration strategy	Estimated run rate	Recommended fidelity level (minimum)
Unique identifier	For example, application ID. Typically available on existent CMDBs or other internal inventories and control systems. Consider creating unique IDs whenever these are not defined in your organization.	R	O	High
Application name	Name by which this application is known to your organization. Include commercial off-the-shelf (COTS) vendor and product name when applicable.	R	R	High
Is COTS?	Yes or No. Whether this is a commercial application or internal development	R	R	High

Attribute name	Description	Discovery, design, and migration strategy	Estimated run rate	Recommended fidelity level (minimum)
COTS product and version	Commercial software product name and version	R	R	High
Description	Primary application function and context	R	O	High
Criticality	For example, strategic or revenue-generating application, or supporting a critical function	R	O	High
Type	For example, database, customer relationship management (CRM), web application, multimedia, IT shared service	R	O	High
Environment	For example, production, pre-production, development, test, sandbox	R	R	High

Attribute name	Description	Discovery, design, and migration strategy	Estimated run rate	Recommended fidelity level (minimum)
Compliance and regulatory	Frameworks applicable to the workload (for example, HIPAA, SOX, PCI-DSS, ISO, SOC, FedRAMP) and regulatory requirements	R	O	High
Dependencies	Upstream and downstream dependencies to internal and external applications or services	R	N/A	High
Infrastructure mapping	Mapping to physical and/or virtual assets that make up the application	R	R	High
License	Commodity software license type (for example, Microsoft SQL Server Enterprise)	R	R	High

Attribute name	Description	Discovery, design, and migration strategy	Estimated run rate	Recommended fidelity level (minimum)
Cost	Costs for software license, software operations, and maintenance	N/A	R	Medium-high
Business unit	For example, marketing, finance, sales	R	O	High
Owner details	Contact information for application owner	R	O	High
Architecture type	For example, web application, 2-tier, 3-tier, microservices, service-oriented architecture (SOA)	R	R	High
Recovery point objective (RPO), recovery time objective (RTO), and /service-level agreement (SLA)	Current service-management attributes	R	R	High

Attribute name	Description	Discovery, design, and migration strategy	Estimated run rate	Recommended fidelity level (minimum)
Revenue-generating Application or business-strategic application?	Yes, if application directly or indirectly influences company revenue or is considered strategic by the business.	R	O	Medium-high
Number of users (concurrent)	For example, internal, or external users or, internal and/or external users/ customers	R	R	Medium-high
User location	Origin of user sessions	R	R	Medium-high
Risks and issues	Known risks and issues	R	O	Medium-high
Migration considerations	Any additional information that could be relevant for migration	R	R	Medium-high
Migration strategy	For example, one of the AWS 6 Rs for migration	R	R	Medium-high

Attribute name	Description	Discovery, design, and migration strategy	Estimated run rate	Recommended fidelity level (minimum)
Database details	For example, partitioning, encryption, replication, extensions, Secure Sockets Layer (SSL) support	R	R	High
Support teams	For example, application operations team name	R	O	Medium-high
Monitoring solution	Product used to monitor this application	R	O	Medium-high
Backup requirements	Required backup schedule in AWS	R	R	Medium-high
DR information	For example, disaster recovery components for this application	R	R	Medium-high
Target AWS requirements	For example, components, account placement, networking, security	R	R	High

Infrastructure

Attribute Name	Description	Discovery, design, and migration strategy	Estimated run rate	Recommended fidelity level (minimum)
Unique identifier	For example, server ID. Typically available on existing CMDBs or other internal inventories and control systems. Consider creating unique IDs whenever these are not defined in your organization.	R	O	High
Network name	Asset name in the network (for example, hostname)	R	O	High
DNS name (fully qualified domain name, or FQDN)	DNS name	O	O	Medium-high
IP address and netmask	Internal and/or public IP addresses	R	R	High
Asset type	For example, physical or virtual server,	R	R	High

	hypervisor, container, device, database instance			
Product name	Commercial vendor and product name (e.g., VMware ESXi, IBM Power Systems, Exadata)	R	R	High
Operating system	For example, REHL 8, Windows Server 2019, AIX 6.1	R	R	High
Configuration	Allocated CPU, number of cores, threads per core, total memory, storage, network cards	R	R	High
Utilization	CPU, memory, and storage peak and average. Database instance throughput.	R	R	High
License	Commodity license type (for example, RHEL Standard)	R	R	High

Is shared infrastructure?	Yes or No to denote infrastructure services that provide shared services such as authentic ation provider, monitoring systems, backup services, and similar services	R	O	High
Application mapping	Applications or application components that run in this infrastructure	R	O	High
Communication data	For example, server to server at a process level	R	N/A	Medium-high
Target AWS requirements	For example, instance types, account, subnets, security groups, routing	R	R	High
Migration strategy, patterns, and tools	For example, one of the 6 Rs for migration, specific technical pattern, migration tooling	R	O	High

Risks and issues	Known risks and issues	R	O	Medium-high

Detailed application assessment

The goal of a detailed application assessment is the complete understanding of the targeted application and its associated infrastructure (compute, storage, and network). High-fidelity data is necessary to avoid pitfalls. For example, it is common for organizations to assume that they fully understand the application. This is natural, and it's true in many cases. However, to minimize risk to the business, it is important to validate institutional knowledge and static documentation by obtaining programmatic data as much as possible. This will take care of the heavy lifting of the discovery process. You can focus on the data elements that come from alternative sources, such as business-specific information, strategic roadmaps, and others.

The key is to avoid last-minute changes during and after migration. For example, when migrating, it's important to avoid changes based on unidentified dependencies that might require the inclusion of a server into an ongoing migration wave. Shortly after migration, it's important to avoid changes based on the associated platform requirements to allow traffic or deploy additional services. These kinds of unplanned changes increase the risk of security and operational issues. We highly recommend using programmatic discovery tooling to validate traffic patterns and dependencies when performing detailed application assessments.

At the beginning of the assessment, you must identify the application stakeholders. These are typically the following:

- Business unit leads
- Application owners
- Architects
- Operations and support
- Cloud-enablement teams
- Specific platform teams such as compute, storage, and networks

There are two approaches for detailed discovery. Top-down discovery starts with the application, or even with the user, and goes all the way down to the infrastructure. This is the recommended approach when the identification of the application is clear. Conversely, bottom-up discovery

starts with the infrastructure and goes all the way up to the application or service and its users. This approach is useful when migration programs are driven by infrastructure teams and when application-to-infrastructure mapping is unclear. In general, you are likely to use a combination of both.

To dive deep into an application, existing architecture diagrams are good start. If these are not available, create one based on current knowledge. Do not underestimate the importance of this task, even for simple rehost or relocate migration strategies. Plotting architectural diagrams helps you to identify inefficiencies that can be quickly addressed with minor changes when in the cloud.

Depending on whether you are performing a top-down or bottom-up approach, the initial diagram will plot application components and services or infrastructure components such as servers and load balancers. After the main components and interfaces have been identified, validate them with programmatic data from discovery tools and application performance monitoring tools. The tools must support dependency analysis and provide communication information between components. Each component that makes up this application must be identified. Next, document dependencies to other applications and services, both internal and external.

In the absence of tooling to validate dependencies and mapping, a manual approach is required. For example, you can log into infrastructure components and run scripts to collect communication information such as open ports and established connections. Likewise, you can identify running processes and installed software. Don't underestimate the effort required for manual discovery. Programmatic tooling can capture and report most dependencies in a few days, except those that occur at greater intervals (typically a small percentage). Manual discovery can take weeks to collect and merge all data points, and it can still be prone to errors and missing data.

Proceed to obtain the information specified in the data requirements section for each prioritized application and the mapped infrastructure. Next, use the following questionnaire to guide you through the detailed assessment process. Meet with the identified stakeholders to discuss the answers to these questions.

General

- What is the criticality level of this application? Is it revenue generating? Is it a business-strategic or supporting-business application? Is it a core infrastructure service shared by other systems?
- Is there any ongoing transformation project for this application?
- Is this an internally or externally facing application?

Architecture

- What is the current architecture type (for example, SOA, microservices, monolith)? How many tiers does the architecture have? Is it tightly coupled or loosely coupled?
- What are the components (for example, compute, databases, remote storage, load balancers, caching services)?
- What are the APIs? Describe these, including API name, operations, URLs, ports, and protocols.
- What is the maximum latency tolerated between components and between this and other applications or services?

Operations

- In what locations does this application operate?
- Who operates the application and infrastructure? Are these operated by internal or AWS Partner teams?
- What happens if this application goes down? Who is affected? What is the impact?
- Where are users or customers located? How do they access the application? What is the number of concurrent users?
- When was the last technology refresh? Is a refresh scheduled in the future? If so, when?
- What are the known risks and issues for this application? What is the history of outages and medium-severity and high severity incidents?
- What is the usage cycle (in business hours)? What is the operating time zone?
- What are the change freeze periods?
- What solution is used to monitor this application?

Performance

- What does the collected performance information show? Is usage spiky or constant and predictable? What is the time frame, interval, and date of the available performance data?
- Are there scheduled batch jobs that are part of or interact with this application?

Software lifecycle

- What is the current rate of change (weekly, monthly, quarterly, or yearly)?
- What is the development lifecycle (for example, test, development, QA, UAT, pre-production, production)?
- What are the deployment methods for application and infrastructure?
- What is the deployment tooling?
- Does this application or infrastructure use continuous integration (CI)/continuous delivery (CD)? What is the level of automation? What are the manual tasks?
- What are the licensing requirements for the application and infrastructure?
- What is the service level agreement (SLA)?
- What are the current test mechanisms? What are the test stages?

Migration

- What are the migration considerations?

At this point, note any considerations when migrating this application. For a more complete and accurate assessment, obtain answers to this question from the different stakeholders. Then contrast their knowledge and opinions.

Resiliency

- What is the current backup method? What products are used for backup? What is the backup schedule? What is the backup retention policy?
- What are the current recovery point objective (RPO) and recovery time objective (RTO)?
- Does this application have a disaster recovery (DR) plan? If so, what is the DR solution?
- When was the last DR test?

Security and compliance

- What are the compliance and regulatory frameworks that apply to this application? What are the last and next audit dates?
- Does this application host sensitive data? What is the data classification?

- Is the data encrypted in transit or at rest, or both? What is the encryption mechanism?
- Does this application use SSL certificates? What is the issuing authority?
- What is the authentication method for users, components, and other applications and services?

Databases

- What databases does this application use?
- What is the typical number of concurrent connections to the database? What are the minimum number and maximum number of connections?
- What is the connection method (for example, JDBC, ODBC)?
- Are connection strings documented? If so, where?
- What are the database schemas?
- Does the database use custom data types?

Dependencies

- What is the dependency between components? Note any dependencies that can't be resolved and that will require migrating the components together.
- Are components split across locations? What is the connectivity between these locations (for example, WAN, VPN)?
- What are the dependencies of this application to other applications or services?
- What are the operational dependencies? For example, maintenance and release cycles such as patching windows.

AWS application design and migration strategy

Designing and documenting the future state of your application is a key migration success factor. We recommend creating a design for any type of migration strategy, no matter how simple or complex. Creating the design will surface potential blockers, dependencies, and opportunities to optimize the application even in cases where the architecture is not expected to change.

We also recommend approaching the future state of the application in AWS with a migration strategy lens. At this stage, make sure that you define what the application will look like in AWS as

a result of this migration. The resulting design will serve as a base for further evolution after the migration.

The following list contains resources to aid the design process:

- AWS Architecture Center combines tools and guidance, such as the AWS Well-Architected Framework. Also, it provides reference architectures that you can use for your application.
- The Amazon Builders' Library contains several resources about how Amazon builds and operates software.
- AWS Solutions Library offers a collection of cloud-based solutions, vetted by AWS, for dozens of technical and business problems. It includes a large collection of reference architectures.
- AWS Prescriptive Guidance provides strategies, guides, and patterns that aid the design process and migration best practices.
- AWS Documentation contains information about AWS services, including User Guides and API References.
- Getting Started Resource Center provides several hands-on tutorials and deep dives to learn the fundamentals so that you can start building in AWS.

Depending on where you are in the cloud journey, AWS foundations might already exist. These AWS foundations include the following:

- AWS Regions have been identified.
- Accounts have been created or can be obtained on demand.
- General networking has been implemented.
- Foundational AWS services have been deployed within the accounts.

Conversely, you might be early in the process, and AWS foundations are not yet established. A lack of established foundations could limit the scope of your application design or require further work to define them. If this is the case, we recommend defining and implementing the foundational design of the landing zone in parallel with the application design work. The application design helps to identify requirements such as AWS account structure, networking, virtual private cloud (VPCs), Classless Inter-Domain Routing (CIDR) ranges, shared services, security, and cloud operations.

AWS Control Tower provides the easiest way to set up and govern a secure, multi-account AWS environment, called a landing zone. AWS Control Tower creates your landing zone using AWS

Organizations, which provides ongoing account management and governance and implementation of AWS best practices based experience working with thousands of customers as they move to the cloud.

Application future state

Start by establishing the initial migration strategy for this application. At this point, the strategy is considered initial because it could change as part of the future state design, which can uncover potential limitations. To validate initial assumptions, see the 6 Rs decision tree. Also, document potential migration phases. For example, will this application be migrated in a single event (all components are migrated at the same time)? Or is this a phased migration (some components are migrated later)?

Note that migration strategies for a given application might not be unique. This is because multiple R types could be used to migrate the application components. For example, the initial approach could be to lift and shift the application without changes. However, an application's components might reside in different infrastructure assets that might require diverse treatments. For example, an application is made up of three components, each running on a separate server, and one of the servers runs a legacy operating system that isn't supported in the cloud. That component will require a replatform approach, while the other two components, running in supported server versions, can be rehosted. It is key to assign a migration strategy to each application component and associated infrastructure that is being migrated.

Next, document the context and problem, and link existing artifacts that define the current state:

- Why is this application being migrated?
- What are the proposed changes?
- What are the benefits?
- Are there any major risks or blockers?
- What are the current downsides?
- What is in scope and out of scope?

Repeatability

Throughout the design work, consider how this solution and architecture for this application can be reused for other applications. Can this solution be generalized?

Requirements

Document the functional and nonfunctional requirements for this application, including security. This includes current and future state requirements, depending on the migration strategy chosen. Use the information gathered during the detailed application assessment to guide this process.

To-be architecture

Describe the future architecture for this application. Consider creating a reusable diagram template that contains building blocks for your source environment (on-premises) and target AWS environment (for example, target AWS Region, account, VPCs, and Availability Zones).

Create a table of components that are being migrated and components that will be new. Include other applications and services (either on premises or in the cloud) that interact with this application.

The following table lists example components. It does not represent a reference architecture or vetted configuration.

Name	Description	Details
Application	External service (inbound connection)	Service consumes data from exposed API.
DNS	Name resolution (internal)	Amazon Route 53 deployed as part of baseline account settings
Application Load Balancer	Distributes traffic among backend services	Replaces on-premises load balancer. Migrate Pool A.
Application security	DdoS protection	Implemented by using AWS Shield
Security group	Virtual firewall	Limit access to applicati on instances on port 443 (inbound).

Name	Description	Details
Server A	Front-end	Rehost, using Amazon Elastic Compute Cloud (Amazon EC2).
Server B	Front-end	Rehost using Amazon EC2.
Server C	Application logic	Rehost using Amazon EC2.
Server D	Application logic	Rehost using Amazon EC2.
Amazon Relational Database Service (Amazon RDS) – Amazon Aurora	Database	Replaces servers E and F
Monitoring and alerting	Change control	Amazon CloudWatch
Audit logging	Change control	AWS CloudTrail
Patching and remote access	Maintenance	AWS Systems Manager
Resource access	Secure access control	AWS Identity and Access Management (IAM)
Authentication	User access	Amazon Cognito
Certificates	SSL/TLS	AWS Certificate Manager
API 1	External API	Amazon API Gateway
Object storage	Image hosting	Amazon Simple Storage Service (Amazon S3)
Credentials	Management and hosting of credentials	AWS Secrets Manager
AWS Lambda function	Retrieval of database credentials and API keys	AWS Lambda
Internet gateway	Outbound internet access	Internet gateway to a VPC

Name	Description	Details
Private subnet 1	Backend and DB	Availability Zone 1 – VPC 1
Private subnet 2	Backend and DB	Availability Zone 2 – VPC 1
Public subnet 1	Front-end	Availability Zone 1 – VPC 1
Public subnet 2	Front-end	Availability Zone 2 – VPC 1
Backup services	Databases and EC2 instance backup	AWS Backup
DR	Amazon EC2 resiliency	AWS Elastic Disaster Recovery

After the components have been identified, plot them in a diagram using your preferred tool. Share the initial design with the key application stakeholders, including application owners, enterprise architects, and the platform and migration teams. Consider asking the following questions:

- Does the team generally agree with the design?
- Can the operations teams support it?
- Can the design be evolved?
- Are there other options?
- Does the design comply with architectural standards and security policies?
- Are any components missing (for example, code repositories, CI/CD tooling, VPC endpoints)?

Architectural decisions

As part of the design process, you will likely find more options for the overall architecture or specific parts of it. Document these options alongside the rationale for a preferred or selected option. These decisions can be documented as architectural decisions.

Ensure that the main options are listed and described with enough detail for a new reader to understand the options and reasons behind a decision to use one option over another.

Software lifecycle environments

Document any changes to current environments. For example, test and development environments will be recreated in AWS and not migrated.

Tagging

Describe mandatory and recommended tagging for each infrastructure component as well as the tagging value for this design.

Migration strategy

By this point of the design, the initial assumptions about migration strategy should be validated. Confirm that there is consensus on the chosen R strategy. Document the overall application migration strategy and the strategies for individual application components. As mentioned previously, different application components might require different R types for migration.

In addition, align the migration strategy to key business drivers and outcomes. Also, describe any phased approach to migration, such as the movement of components in different migration events.

For more information about determining your 6 Rs, see the AWS Migration Hub strategy recommendations.

Migration patterns and tools

With a defined migration strategy for the application and infrastructure components, you can now explore specific technical patterns. For example, a rehost strategy can be implemented by migration tooling such as AWS Application Migration Service. If you don't need to replicate the state or data, you can achieve the same outcome by redeploying the application using an Amazon Machine Image (AMI) and an application deployment pipeline.

Similarly, to replatform or refactor (re-architect) an application, you can use tools such as AWS App2Container, AWS Database Migration Service (AWS DMS), AWS Schema Conversion Tool (AWS SCT), AWS DataSync. For containerizing, you can use Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), or AWS Fargate. When repurchasing, you can use an AMI for a specific product or a software as a service (SaaS) solution from AWS Marketplace.

Evaluate the different patterns and options that are available for achieving the goal. Consider pros and cons, and migration operational readiness. To help with your analysis, use the following questions:

- Can migration teams support these patterns?
- What is the balance between cost and benefits?
- Can this application, service, or component be moved to a managed service?
- What is the effort to implement this pattern?
- Is there any regulation or compliance policy preventing the use of a specific pattern?
- Can this pattern be reused? Reusable patterns are preferred. However, sometimes a pattern will be used only one time. Consider balance between the effort of a single-use pattern over an alternative reusable pattern.

AWS Prescriptive Guidance contains a variety of migration patterns and techniques.

Service management and operations

When creating application designs for migration to AWS, consider operational readiness. When evaluating readiness requirements with your application and infrastructure teams, consider the following questions:

- Are they ready to operate it?
- Are incident response procedures defined?
- What is the expected service level agreement (SLA)?
- Is separation of duty required?
- Are the different teams ready to coordinate support actions?
- Who is responsible for what?

Cutover considerations

Considering the migration strategy and patterns, what is important to know at the moment the application is migrated? Cutover planning is a post-design activity. However, document any considerations for activities and requirements that can be anticipated. For example, document the requirement to perform a proof of concept, if applicable, and outline the test, audit, or validation requirements.

Risks, assumptions, issues, and dependencies

Document any open risks, assumptions, and potential issues that are not yet resolved. Assign clear ownership to these items, and track progress so that the overall design and strategy can be approved for implementation. In addition, document key dependencies for implementing this design.

Estimating run cost

To estimate the cost of your target AWS architecture, use the AWS Pricing Calculator. Add your infrastructure components as defined by your design, and obtain an estimated run cost. Factor in software licenses that are required for your application components and that are not already included in the AWS services that you will use.

Portfolio analysis and migration planning

This stage of assessment focuses on completing the portfolio-level discovery and analysis started in the Portfolio discovery and initial planning section. The goal is to iterate and establish a baseline for the initial portfolio of applications and infrastructure. This baseline includes identifying all dependencies, iterating rationalization models for migration, creating a detailed business case, and outlining a migration wave plan. As a result, the required data fidelity is higher. This stage will require time investment. To accelerate assessment outcomes, we recommend using as many programmatic data sources, such as discovery tooling, as possible.

Primary outcomes of this stage include the following:

- A high-fidelity application and infrastructure inventory
- A high-level migration strategy for each application
- A high-confidence migration wave plan
- A detailed business case

Understanding complete assessment data requirements

The following table describes the information required to obtain a complete portfolio view of the applications in the migration and their associated infrastructure.

The tables use the following abbreviations:

- R, for required
- O, for optional
- N/A, for not applicable

Applications

Attribute name	Description	Inventory and prioritization	Detailed Business case	Recommended fidelity level (minimum)
Unique identifier	For example, application	R	R	High

Attribute name	Description	Inventory and prioritization	Detailed Business case	Recommended fidelity level (minimum)
	ID. Typically available on existent CMDBs or other internal inventories and control systems. Consider creating unique IDs whenever these are not defined in your organization.			
Application name	Name by which this applicati on is known to your organizat ion. Include commercial off-the-shelf (COTS) vendor and product name when applicable.	R	R	High
Is COTS?	Yes or No. Whether this is a commercia l application or internal development	R	R	High

Attribute name	Description	Inventory and prioritization	Detailed Business case	Recommended fidelity level (minimum)
COTS product and version	Commercia l software product name and version	R	R	High
Description	Primary applicati on function and context	R	R	High
Criticality	For example, strategic or revenue-g enerating application, or supporting a critical function	R	R	High
Type	For example, database, customer relationship managemen t (CRM), web application, multimedia, IT shared service	R	R	High
Environment	For example, production, pre-production, development, test, sandbox	R	R	High

Attribute name	Description	Inventory and prioritization	Detailed Business case	Recommended fidelity level (minimum)
Compliance and regulatory	Frameworks applicable to the workload (for example, HIPAA, SOX, PCI-DSS, ISO, SOC, FedRAMP) and regulatory requirements	R	R	High
Dependencies	Upstream and downstream dependencies to internal and external applications or services. Non-technical dependencies such as operational elements (for example, maintenance cycles).	R	O	High
Infrastructure mapping	Mapping to physical and/or virtual assets that make up the application	R	R	High

Attribute name	Description	Inventory and prioritization	Detailed Business case	Recommended fidelity level (minimum)
License	Commodity software license type (for example, Microsoft SQL Server Enterprise)	R	R	Medium-high
Cost	Costs for software license, software operations, and maintenance	N/A	R	Medium-high
Business unit	For example, marketing, finance, sales	R	R	High
Owner details	Contact information for application owner	R	R	High
DR information	Disaster recovery components	R	R	High
Migration strategy	For example, one of the 6 Rs for migration to AWS	R	R	High

Attribute name	Description	Inventory and prioritization	Detailed Business case	Recommended fidelity level (minimum)
Support tickets	12–24 months of data to help assess the productivity and financial impact of outages, slow downs, transaction throttling, and batch window overruns	O	R	Medium

Infrastructure

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Unique identifier	For example, server ID. Typically available on existing CMDBs or other internal inventories and control systems. Consider creating unique IDs whenever these are not defined in your organization.	R	R	High

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Network name	Asset name in the network (e.g., hostname)	R	R	High
DNS name (fully qualified domain name, or FQDN)	DNS name	R	O	High
IP address and netmask	Internal and/or public IP addresses	R	R	High
Asset type	For example, physical or virtual server, hypervisor, container, device, database instance	R	R	High
Product name	Commercial vendor and product name (for example, VMware ESXi, IBM Power Systems, Exadata)	R	R	High
Operating system	For example, REHL 8, Windows Server 2019, AIX 6.1	R	R	High

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Configuration	Allocated CPU, number of cores, threads per core, total memory, storage, network cards	R	R	High
Utilization	CPU, memory, and storage peak and average. Database instance throughput.	R	R	High
License	Commodity license type (for example, RHEL Standard)	R	R	High
Is shared infrastructure?	Yes or No to denote infrastructure services that provide shared services such as authentic ation provider, monitoring systems, backup services, and similar services	R	R	High

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Application mapping	Applications or application components that run in this infrastructure	R	R	High
Cost	Fully loaded costs for bare-metal servers, including hardware, maintenance, operations, storage (SAN, NAS, Object), operating system license, share of rack space, and data center overheads	N/A	R	Medium-high
Estimated volume of data transfer (in/out)	For example, per infrastructure asset over per day over a 30 days period	O	R	Medium

Networks

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Size of pipe (Mb/s), redundancy (Y/N)	Current WAN link specifications (for example, 1000 Mb/s redundant)	R	R	Medium-high
Link utilization	Peak and average utilization, outbound data transfer (GB/month)	R	R	Medium-high
Latency (ms)	Current latency between connected locations.	R	O	High
Cost	Current cost per month	N/A	R	Medium-high

Migration

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Rehost	Customer and partner effort for each workload (person-days), customer and Partner	N/A	R	Medium-high

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
	cost rates per day, tool cost, number of workloads			
Replatform	Customer and partner effort for each workload (person-days), customer and partner cost rates per day, number of workloads	N/A	R	Medium-high
Refactor	Customer and partner effort for each workload (person-days), customer and partner cost rates per day, number of workloads	N/A	R	Medium-high
Retire	Number of servers, average decommission cost	N/A	R	Medium-high

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
Landing zone	Re-use existing (Y/N), list of AWS Regions needed, cost	N/A	R	Medium-high
People and change	Number of staff to train in cloud operations and development, cost of training per person, cost of training time per person	N/A	R	Medium-high
Duration	Duration of in-scope workload migration (months)	O	R	Medium-high
Parallel cost	Time frame and rate at which as-is costs can be removed during migration	N/A	R	Medium-high

Attribute name	Description	Inventory and prioritization	Business case	Recommended fidelity level (minimum)
	Time frame and rate at which AWS products and services, and other infrastru cture costs, are introduced during migration	N/A	R	Medium-high

Establishing a baseline for the application portfolio

To create high-confidence migration wave plans, you must establish a baseline for the portfolio of applications and its associated infrastructure. A portfolio baseline provides a comprehensive view of the migration scope, including technical dependencies and migration strategy. The portfolio baseline provides clarity about which applications are in-scope of migration and that the data points outlined in the Understanding complete assessment data requirements section are collected. Likewise, all the associated infrastructure (compute, storage networks) is understood and mapped to the applications.

Technical dependencies can be described in four categories:

- **Application-to-infrastructure** dependencies establish the link between software and physical or virtual hardware. For example, there is a dependency between a CRM application and the virtual machines where it is installed.
- **Application-component** dependencies describe how components running in different infrastructure assets interact. An example of an application-component dependency is a web front end running on virtual machines, with an application layer running on a different virtual machine, and a database running on a database cluster.
- **Application-to-application** dependencies relate to the interaction between applications or application components with other applications or their components. An example of an application-to-application dependency is a payment-processing application and a stock-

management application. These applications are independent, but they constantly interact using defined API operations.

- **Application-to-infrastructure services** dependencies are technically application-to-application dependencies, given that the infrastructure service is itself an application. However, we recommend categorizing these separately. The main reason is that infrastructure services typically are shared by many applications, so they have a long trail of dependencies. They also typically follow a different migration strategy and pattern.For example, a load balancer can contain balancing pools for several applications. What matters is the dependency to the pool, which is likely to be migrated individually, alongside the dependent application, while the load balancer itself is retained or retired.In addition, individualizing application-to-infrastructure service dependencies helps to avoid false dependency groups. A false dependency group is when several business applications are grouped together, implying that have a common dependency to an infrastructure service must be migrated at the same time. For example, authentication services, such as Active Directory, are likely to be associated with large groups of applications. The key is to approach these applications individually and to address the dependency by enabling the service, such as AWS Directory Service for Microsoft Active Directory, in the cloud environment.

When you establish a baseline for the portfolio, we recommend that you confirm a migration strategy for each application component. The migration strategy will be one of the 6 Rs for migration (see the Iterating the 6 Rs migration strategy section). In the portfolio baseline, one of the 6 Rs should be associated with each application. A 6 R strategy should also be associated with each of the application's infrastructure components.

To establish a baseline version of the portfolio, including dependencies and migration strategies, use automated discovery tooling (see Evaluating the need for discovery tooling). Complement the data with information gathered from key stakeholders such as application owners and infrastructure teams. Keep gathering data until you obtain a complete portfolio inventory that matches the attributes and level of fidelity outlined in the data requirements section for this stage. The resulting dataset will be instrumental in driving the migration.

Consider that, depending on the extent of your migration scope and the available tooling, this activity can take several weeks to complete.

Iterating the prioritization criteria

Before you create migration wave plans, we recommend that you iterate the application prioritization criteria to pivot from pilot application selection to long-term wave planning.

In earlier sections, we introduced a default prioritization criteria that would prioritize simple cloud-ready applications (see Prioritizing applications). This was because in early stages we recommend starting with noncritical applications to refine migration processes and incorporate lessons learned. However, at this stage, and to create long-term plans, the order in which applications are migrated should be aligned to business drivers. Applying the new criteria will generate a new ranking of applications that will be a key input for wave planning.

Review the available data points from the application portfolio, and select the attributes that will determine application prioritization based on business drivers.

First, validate your business drivers (see Business drivers and technical guiding principles). Next, based on your business drivers, select the attributes that will help to prioritize applications for migration.

The following table shows example prioritization criteria aligned to business drivers for innovation.

Attribute or data point	Possible values	Score (0-99)	Importance or relevance multiplying factor
Operating system	AIX	80	High (1x)
	Solaris	80	
	HP-UX	80	
	Mainframe	70	
	Windows	50	
	Linux	20	
Business criticality	High	60	High (1x)
	Medium	40	

Attribute or data point	Possible values	Score (0-99)	Importance or relevance multiplying factor
	Low	20	
Architecture	Tightly coupled	60	High (1x)
	Loosely coupled	20	
Operating model	Traditional - no CI/CD	60	Medium-high (0.8x)
	Basic CI/CD	40	
	Full DevOps	20	
Number of compute instances	1-3	60	Medium-high (0.8x)
	4-10	40	
	11 or more	20	
Migration strategy	Refactor (re-architect)	70	Medium (0.6x)
	Replatform	40	
	Repurchase	30	
	Rehost	10	

The following table shows example prioritization criteria aligned to business drivers for quick cost reduction.

Attribute or data point	Possible values	Score (0-99)	Importance or relevance multiplying factor
Database product	Oracle	70	High (1x)

Attribute or data point	Possible values	Score (0-99)	Importance or relevance multiplying factor
	Microsoft SQL	70	
	Others	20	
Operating system	Windows	70	High (1x)
	Linux	70	
	Others	20	
CPU utilization (average)	More than 36%	60	High (1x)
	Less than 36%	40	
Number of compute instances	11 or more	60	Medium-high (0.8x)
	4-10	40	
	1-3	20	
Migration Strategy	Retire	80	Medium (0.6x)
	Rehost	70	
	Replatform	50	
	Refactor (re-architect)	10	

Test the prioritization criteria and iterate until you generally agree with the output. It takes at least three or four iterations to obtain a baseline version.

Iterating the 6 Rs migration strategy selection

At this stage, we recommend that you iterate and evolve the 6 Rs decision tree. The Determining the R type for migration section introduced a default decision tree. We recommend revising the

tree, considering the learnings throughout migration of the initial pilot applications, and ensuring that it still aligns to business drivers, prioritization criteria, and your unique circumstances. Validate the decision tree with sample applications, and verify that it still produces the expected strategy. Otherwise, update the logic accordingly. The resulting tree will be key in establishing baselines for the portfolio of applications and in allocating migration strategies for each application component.

As described in the previous 6 Rs section, the 6 Rs also apply to infrastructure, and it is equally important to assign them accordingly. While a given application component will have a migration strategy, at an infrastructure level, each infrastructure asset will follow a given migration strategy that might be different from the strategy established for the application component it supports.

Remember that the 6 Rs decision tree applies to application components only. The migration strategy for infrastructure is derived from the strategy chosen for the application. For example, for an application component that will be replatformed, the current infrastructure that hosts it could be retired.

Ensure that migration strategies are allocated to each application component and its associated infrastructure. This information will be a key factor when estimating effort, capacity, and skills needed, and when creating migration wave plans.

For more information about determining your 6 Rs, see the AWS Migration Hub strategy recommendations.

Wave planning

In wave planning, a dependency group is a collection of applications and infrastructure that have technical and nontechnical dependencies that can't be resolved. Because of these dependencies the applications and infrastructure in a dependency group must be migrated at the same time or on a specific date. For example, an application running on a virtual machine and a database running in a separate virtual machine, where there are low-latency requirements or high-traffic volumes and complex queries, are likely to be migrated together rather than operating one component in the cloud and the other on premises. Likewise, independent applications that interact over an API with similar low-latency requirements will also be migrated at the same time.

Migration waves typically span 4–8 weeks, and they can contain one or more migration events. Dependency groups are combined into waves so that a wave can contain one or more dependency groups. The wave also contains other activities that are required for the migration. These include AWS infrastructure setup (such as landing zone, security, and operations), migration tooling,

and migration activities such as data replication, cut-over planning, testing, and post-migration support.

To measure success and track progress, waves should be aligned to outcomes and business drivers. This will also influence wave duration and the dependency groups that a wave contains. The completion of a wave should reflect a measurable achievement. The planning of a wave can also combine other factors, such as technical guiding principles. For example, waves can be defined by environment (for example, development, test, production) or by migration strategy (for example, rehost wave, replatform wave).

To create effective and high-confidence migration wave plans, you must obtain a complete view of the application portfolio, associated infrastructure (compute, storage, networks), dependency mapping, and migration strategy.

The section on establishing a baseline for the application portfolio described four categories of technical dependencies. These dependencies contribute to the creation of migration waves and the definition of dependency groups. Dependency groups will be determined by the criticality of the dependency. In addition, nontechnical dependencies must be considered. For example, application release schedules, maintenance windows, and key business dates such as end of month of end of quarter processing will influence the wave plan.

Determine whether the dependency is *soft* or *hard*. A soft dependency is a relationship between two or more assets, or from an asset to a constraint, that is not dependent on the location of the components. For example, two systems that operate in the same local network (or same infrastructure) can be split apart by moving one of those systems to the cloud while the other remains on premises. Another example is a system that can be migrated during a maintenance window without impacting maintenance activities.

A hard dependency is a relationship between two or more assets, or from an asset to a constraint, that is dependent on location. For example, two systems that operate in the same local network, and that are heavily dependent on low latency for communication between the application server and the database server, have a hard dependency. Moving only one of these systems to the cloud would cause functionality or performance problems that cannot be resolved. Likewise, nontechnical reasons, such as resource availability (for example, the team performing the migration) or operational constraints, such as maintenance windows where two systems can only be migrated in a given time window, might create a hard dependency for these assets.

To create a migration wave plan, determine your dependency groups by analyzing dependencies, ideally from a highly trusted source of data such as specialized discovery tooling, and combine

this information with your application prioritization criteria and operational circumstances. The applications at the top of the prioritization ranking should be targeted for your initial migration waves. Determine wave capacity (the number of applications a wave can contain) based on resource availability, risk tolerance, business and technical constraints, experience, and skills. Consider working with AWS Professional Services or AWS Migration Competency Partners, that can provide specialists to assist you throughout the process.

The prioritization criteria are an initial indication of the order in which you will move your applications to the cloud. However, dependency groups will be the actual determinant for the applications that will be moved at a given time. This is because applications that are ranked as high priority could have hard dependencies to applications that are in the middle or at the bottom of the ranking.

The migration strategy will also influence the composition of a wave. For example, a high-priority application that requires a refactor strategy that might require several weeks or months of analysis, design, testing, and preparations will likely be placed in a later wave.

Creating a wave plan

A prerequisite to migrating a wave of applications is the application portfolio data and the detailed application assessment of the group of applications that will be migrated in the wave. The detailed assessment should include the list of applications in the wave, the associated infrastructure details, a target design, and a migration strategy for each application.

Establishing wave ownership and governance is key to managing and tracking the wave work, program dependencies, change management, issues, and risks. Ensure that a governance framework is in place to manage the plan.

To outline the wave plan, start with a default wave construct. What happens within a wave? After the initial input is defined, the wave can commence. Typically, the activities will be:

1. Refine the cutover plan. This activity should outline the runbooks and steps that must be taken at the moment of migration, including the coordination with other internal and external teams.
2. Refine the rollback plan. What must be done to roll back the applications if things go wrong?
3. Prepare the target infrastructure. For example, you can create or extend the AWS landing zone (AWS account, security, networking, infrastructure services, other supporting infrastructure).
4. Test target infrastructure.
5. Operate migration tooling. For example, install replication agents and start data transfer.

6. Conduct cutover plan and runbook dry runs. Group all participating team members, and review all steps in advance.
7. Monitor data replication and infrastructure deployments.
8. Confirm readiness for operation of infrastructure and applications in AWS.
9. Confirm security readiness.
10. Confirm compliance and regulatory requirements (for example, workload validation pre-migration and post-migration) if applicable.
11. Migrate the applications to AWS and perform pre go-live testing.
12. Provide post-migration support for a period of time, such as 3 days, when the operations teams and the migration teams are fully available to resolve issues, and apply optimizations.
13. Conduct a post-migration review. Document lessons learned, and incorporate them into future waves.
14. Perform wave closure by confirming operational handover and obtainment of metrics for reporting.

How long each of these activities takes will be dictated by the complexity of the scope, the wave capacity, the people involved, and your unique circumstances. Where possible, smaller waves are preferable because that will reduce the impact of any delays or migration blockers. Determine, with your teams, what the default duration of a wave will be.

Next, proceed to analyze dates to create an initial high-level structure of empty waves (with no application assigned yet). Consider the following questions:

- What is the total migration program length?
- What are the deadlines?
- Are there fixed data center exit dates?
- Are there collocation contract end dates?
- What are the application and infrastructure refresh cycles?
- What are the application maintenance and release cycles?
- Are there any dates when migrations should be avoided (for example, release and maintenance cycles, end of year, holidays, month-end processing)?

With these considerations, plot the waves into a plan. To accelerate the migration process, we recommend overlapping waves where possible. The key to overlapping waves is to define and

consider what happens within a wave. Typically, deployment activities, target infrastructure validation, and data synchronization will occur during the first half of a wave. The second half will focus on the actual migration, testing, and operational handover. This means that different teams are involved in each half of the process, and that you can gain some efficiencies. For example, as soon as the team involved in target infrastructure preparation has completed their work, they can start working on the requirements of the next wave. In general, it is preferable that most waves have a similar length and structure to facilitate a factory-like approach to migrations. However, during the wave planning process, the size of a given wave can be extended to meet dependencies or operational requirements.

Next, based on the dependency groups that have been identified, determine the maximum size of a wave in terms of the number of dependency groups it can contain. Wave size is typically dictated by risk appetite (for example, how much parallel change can be tolerated), and resource availability (for example, how much parallel change can be performed with the available resources, skills, and budget). However, during early planning, don't be limited by resource requirements and availability. Waves that contain more than one dependency group can be decomposed into smaller waves in future iterations.

After the dependency groups for a given wave have been confirmed, review resource requirements for migrating the wave. Consider adjusting the wave size (the number of dependency groups it contains) based on resource requirements. This might lead to smaller or larger waves. Iterate the wave plan as needed until all waves have been defined.

Managing change

The portfolio of applications and associated infrastructure will change during the lifecycle of migration programs. Long-running migration programs coexist with normal business evolution and change. Applications keep evolving as they wait to be migrated. Servers are added or removed, new infrastructure is deployed on premises. It is expected that the scope of a wave or dependency group will require changes. Changes are required especially when, closer to the migration date, a previously unknown dependency is identified, or a new server is included in the inventory. Sometimes this can happen during the migration itself.

Scope changes affect dependency groups and waves. To handle change and minimize impact, it's important to establish a scope-control mechanism. A scope change control mechanism requires the definition of a single source of truth for the scope. This could be a tool for managing the scope, or a .csv file, spreadsheet, or database, as defined by the migration program governance. You must

identify changes, analyze impact, and communicate change to the relevant stakeholders so that they can take action. The wave plan will be iterated as a result.

Detailed business case

In this stage, we recommend validating and expanding the scope of the business case to provide a greater level of detail to support the transformation program. The quickly assembled initial directional business case is designed to provide enough confidence to invest in the foundational steps and next level of detailed planning.

Developing a detailed business case supports this planning process in the following ways:

- Providing financial analyses that inform decisions on what should be migrated and modernized, which options to select and how to phase and prioritize the work
- Validating, refining, and developing the original directional financial case by re-examining in detail:
  - The infrastructure cost-reduction potential
  - The internal IT productivity and any outsourced operations efficiencies
  - The estimates for the investments needed for program setup, migration, and modernization
- Identifying, estimating the scale of, and setting up the process for tracking the further value drivers that migration brings

In the detailed business case, you establish the following:

- The objective basis on which to secure the mandate and investment to implement at least the first phase of migration
- The baseline minimum financial performance expectation for the program
- Clarity over the financial basis on which various migration design and prioritization decisions are made, so that when circumstances and people change over the course of the program, the new leadership can make informed choices.
- Insight into incremental areas of cost optimization to be explored after initial usage data becomes available as workloads are migrated and start operation
- Estimates for the value that cloud transformation brings to the business from increased resilience and agility

- The associated KPIs, metrics, and assumptions used to estimate the financial return from improved resilience and agility, which then form the baseline for driving the primary benefits realization out of the program

Determine the scenarios needed for the case

When building the detailed business case, it is usually necessary to develop multiple scenarios to support the various purposes that the business case is used for.

Minimum change scenario – To assess the minimum financial performance expectation, prepare a scenario that assumes the minimum expected change to the status quo. This scenario, as a worst-case scenario, is useful support when getting the mandate to invest in the migration. This scenario models the minimum expected degree of capacity growth and minimum changes for other quality-of-service needs, such as availability and resilience. The least change creates the lowest cost and least resource inefficiencies for the current operating model.

Most likely scenario – To inform program strategy and prioritization decisions, prepare the scenario that reflects what the business expects to happen. This scenario should include the likely peak utilization growth or reduction and the upgrade costs to meet demand for high levels of service quality (especially availability and resilience) from the business.

Other specific scenarios – Where it is still necessary to make an assumption that could have a large impact on the business case, develop scenarios for both where the assumption holds true and where it does not. However, we recommend keeping the number of these alternative scenarios to the absolute minimum. Creating any more than three to four scenarios in total slows progress, and becomes expensive, confusing and difficult to maintain. Wherever possible, conduct experiments and work to remove larger assumptions.

Validate and refine the infrastructure and migration cost model

After you have completed the portfolio analysis and prepared the design and sizing of the target AWS services, refine the running cost estimates for the current operating model (COM) and future operating model (FOM) on AWS for each scenario. It is usually necessary to refine the estimates for the following:

- **COM infrastructure costs** of hypervisor host server, bare-metal server, storage, network device, security appliance hardware refreshes, installation, and maintenance. Calculate these with actual pricing and discount levels for the capacity needed for the scenario.

- **COM data center and collocated facilities costs**, including space, cooling, power, racks, uninterruptible power supply (UPS), cabling, physical security systems, sized for the growth and specified to meet the capacity, and high availability and disaster recovery (DR) levels for the scenario.
- **COM network services costs**, including costs for WAN links, content delivery networks, and virtual private networks (VPNs), calculated using contracted pricing for the connectivity, bandwidth, throughput, and latency needs for the scenario.
- **COM application and infrastructure software costs** based on existing contracts to provide the growth or reduction of usage for the scenario.
- **FOM AWS utility costs**, including tech support and managed services as needed, based on the refined service architecture, instance sizes, preferred pricing model, expected usage, and usage volatility.
- **FOM application licensing** based on final application design, the configuration of the infrastructure running the applications, growth over time, and license transferability rules.
- **FOM migration and modernization cost estimates**, refined to reflect the baseline migration wave plan for the scenario, and detailed to provide costs for each workload, especially for those to be replatformed, repurchased, or refactored.
- **FOM decommissioning costs**, including estimates of asset write-off and contract early termination costs, revised to reflect decommissioning timing in the baseline migration wave plan, verification of what assets can be repurposed and what assets can be switched around to minimize write-offs, and the cost of disposal of the physical assets and media.
- **Migration parallel run costs** refined to reflect the timing of each migration cutover and each existing service decommissioning.

Refine the IT productivity and IT operations and support efficiency value model

As with the directional business case, there are two primary approaches to refining and developing the value model around IT operations and support. The approach that you choose depends on whether the COM is managed in-house or with contractors or outsourced services:

Internal team productivity improvement

Where IT operations and support are managed in house, the focus of the business case is on the following:

- Identifying and quantifying the productivity gains from migration and any operational automation that is included in scope
- Validating that the time freed up for the in-house team can be readily and productively applied to other typically higher-value activities, giving opportunities for progression and greater reward to the team and more value to the organization

Assess how much time each member in each role within the team spends on their various regular activities, and guidance on the expected reduction in workload for different activities.

The following table provides initial guidance for the typical levels of workload reduction by activity for those tasks that consume the bulk of IT operations and support effort across the different roles in the team. The table includes a description of how the productivity is achieved.

> **Note**
>
> The activities listed are typically performed by team members in several different roles, so the productivity saving for each task should be assessed across the full set of roles in the team. For example, in IT operations teams organized by infrastructure tower (such as compute, storage, and networking), capital expenditure planning and budgeting might be common to tower leads for each tower.

Operational and support activities	Level of savings	Productivity driver
Infrastructure design	Medium	Design is simplified, with fewer parameters to be considered.
Capital expenditure planning and budgeting	High	OPEX-centric elastic services remove virtually all budgeting and planning issues.
Purchasing	High	Procurement is greatly simplified after the AWS accounts are established.

Operational and support activities	Level of savings	Productivity driver
Capacity planning	Medium-very high	Networking and compute capacity management workload is usually all but eliminated, and for storage it is heavily simplified
Tuning	High-very high	Tuning is not needed for managed services and barely needed for other services because instances can be changed in size at any time.
Managing hardware failure	Very high	All aspects of handling hardware in the cloud are handled transparently by AWS.
Monitoring server availability and communications	High	Monitoring and communica tions are simplified extensive ly with AWS tool support and automation.
Security management	Medium	Workload is significantly reduced with AWS security capabilities and with AWS owning the security responsibilities for the AWS Cloud hardware, software, networking and facilities.
Network and storage upgrades, maintenance, and patches.	Very high	All aspects of network and storage maintenance in the cloud are handled transpare ntly by AWS.

Operational and support activities	Level of savings	Productivity driver
Racking and stacking – hardware logistics	Very high	All aspects of managing hardware in the cloud are handled transparently by AWS.
Backup	Medium	Backup is simplified extensive ly with AWS tools, flexible storage systems, and automation.
Managed services (such as Amazon S3, Amazon RDS, AWS Lambda, and AWS Fargate)	Very high	Managed services run on environments that are fully managed by AWS, so they require no maintenance, patching, monitoring, or provisioning management activity.
Device and service setup and commissioning	High-very high	Activities for hardware setup for the estate migrated to AWS are usually reduced, except for WAN connectiv ity devices for establishing VPNs or AWS Direct Connect connections to AWS data centers.
Endpoint protection and antivirus protection	High	Application and maintenance of endpoint protection and antivirus services is typically extensively automated as part of the migration design.

Operational and support activities	Level of savings	Productivity driver
Threat, vulnerability, and risk assessments	High	AWS provides support for the elements of this, focused on the core platform and the mechanisms that AWS provides to secure architect ures simplifies assessment.
Data center infrastructure project management	High	Project management for installation work for expansion, refresh, or decommissioning of infrastru cture services. While some management of infrastru cture software and services remains, this is much simpler than on-premises infrastru cture, and the hardware activities are eliminated.
Data center facilities management	Medium-very high	The facilities managemen t work attributable to all the servers, storage devices, security appliance s, and associated racks is removed for everything that is migrated. However, some work usually remains for providing facilities for WAN link network devices and for any infrastructure that is kept on premises in a hybrid architecture.

Operational and support activities	Level of savings	Productivity driver
Application architecture, development, management, and testing	Low	Use of agile development tool-chains, in combinati on with automation of application stack instantia tion and destruction for building test environments as needed, reduces application development lead times and eliminates many manual test steps.
Installing and configuring application software	Medium	Full application stack installation and configura tion is readily automated using services such as AWS CloudFormation and simplifie d through the use of landing zones, which can be readily configured by using AWS Control Tower.
IT support	Medium	Reductions in L1 and L2 support are achieved by reducing capacity and performance issues through the use of Service Catalog capabilities for self-service provisioning, increased use of low-cost high availabil ity architectures (reducing outages and configuring automatic scaling and edge computing).

Operational and support activities	Level of savings	Productivity driver
Database administration	Minimal-low	These activities remain mostly unchanged. They are typically resourced at the same levels for AWS as for on-premises infrastructures.
Infrastructure and security requirements capture, analysis, and design	Minimal	
Documentation	Minimal	
Application and performance monitoring	Minimal	
L3 technical support, answering queries, and troubleshooting and problem solving	Minimal	
Installing and configuring application software	Minimal	
Application L3 support (excluding budgeting and long-range capacity planning)	Minimal	

The following table shows the expected savings for each level of workload reduction.

Level	Expected
Very high	85% - 100%
High	60% - 90%
Medium	30% - 70%
Low	10% - 35%
Minimal	0% - 10%

These metrics provide a starting point for assessing productivity gains and including them in the detailed business case. Actual productivity gains vary based on the specific situation. It can be useful to calculate the productivity savings at both the midpoint and lower end of the ranges to estimate typical and conservative scenarios.

As the program progresses, it is valuable to capture actual data for time spent on each activity by role. That data builds an improved base for estimating operations and supports costs for new projects and expansions of services.

Outsourced IT operations and support cost reduction

Where IT operations and support are primarily outsourced or managed with contractors, the cost allocation for the future operating model (FOM) can be prepared by requesting quotations from AWS Partners that offer managed service solutions, including AWS Partner-led AWS Managed Services (AMS). You can also contact your AWS account manager and request a price for AMS directly, as described in the subsection on Building in operational cost optimization within the Creating a directional business case section.

For the detailed business case, replace any benchmark figure with a quotation based on the revised AWS services bill of materials and expected service consumption, the AMS package and any options needed, and the service level needed. The cost will have a one-time implementation component and a consumption-based run rate.

Include any remaining IT operations, support that must be retained for any service that will not be migrated to AWS, and a one-time cost if there are any contract penalties (for example, for early termination).

Develop the resilience value model

On AWS, you can construct a wide range of high availability, disaster recovery, and fault-tolerant architectures. Consumption-based pricing means that services are charged for only when used. Together, these two factors provide exceptional cost performance for resilience.

Furthermore, AWS ccustomers have been using this to improve their workloads' resilience. The IDC 2018 survey gives examples of participating customers achieving 73 percent fewer outages per year, a 58 percent reduction in mean time to recover (MTTR) and a 94 percent reduction in lost productivity. The same survey showed that the financial benefits derived through increased resilience were 50 percent greater than the IT infrastructure cost reduction benefit.

In addition, further resilience is achieved through modernizing the software development lifecycle for applications. Where CI/CD pipelines with test automation are introduced to support greater business agility, software defects are caught earlier in the development cycle, greatly reducing software maintenance costs.

To assess and include this value in the business case, first work with application business owners to build a picture of the *total benefit opportunity* for each workload to be migrated**.** This might include as the following items:

- The number, average duration, and nature of interruptions in service:
  - Examples of service interruptions include outages, performance slow-downs, planned batch and maintenance windows overrunning, bugs in key functions, and access throttling during peak periods.
- Impact on revenue by interruptions of revenue-generating services, such as ecommerce systems:
  - The likely number of transactions unable to be completed through service interruptions, based on the interruption time and transaction rates
  - Average value for each transaction impacted
- The additional cost of support engineers' time to resolve defects in production systems compared to the cost of discovering them earlier in the development process
- Impact on internal users' productivity and the cost of lost time

Then make an assessment of an expected and a more conservative reduction in time lost to service interruptions that the increased resilience should yield. For example, consider including the following items:

- Reduced number of outages and MTTR using high availability architectures and improved recovery time objective (RTO) and recovery point objective (RPO)
- Reduction in slow-downs, elimination of capacity throttling and avoidance in batch processing overruns, using capabilities such as automatic scaling
- Reduced number of application bugs that are discovered only in production, through the implementation of CI/CD pipelines and automated regression testing on infrastructure spun up and spun down to minimize cost

Put these together for the portfolio of applications to be migrated and modernized, and calculate the expected and more conservative business value figures for each year of the case. The benefits

should ramp up in line with the migration schedule and then scale in volume in line with the usage growth expectations of the contributing applications.

Develop the business agility value model

Business agility is the prime reason that AWS customers migrate to AWS. The IDC 2018 Survey of AWS customers indicated that for them, business agility benefits accounted for 47 percent of the total benefits measured and over five times the benefits accruing from infrastructure cost reduction.

Accurately predicting all the business agility benefits that will accrue from any transformation is challenging. However, by focusing on applications that support large numbers of users or are sources of business differentiation, you can model and include a material portion of this benefit into the baseline detailed business case.

As the migration proceeds, incrementally refine and expand the business agility value model as more benefits become quantifiable. This keeps the business case relevant, so that it can be used as the primary decision support tool with which to steer the program.

To build the business agility value model, use the following guidance:

- Select the workloads that have the opportunity to drive the greatest business performance improvement, such as:
  - Revenue-generating workloads
  - Business operations workloads with scope for driving efficiency gains and removing costs from the business
  - Business productivity tools supporting large user bases
- For revenue and efficiency generating workloads, do the following:
  - Make a realistic and a more conservative assessment of the revenue growth or operational efficiencies that major and minor application upgrades could be expected to drive.
  - Estimate the increased number of major and minor releases per year that AWS increased application development speed and reduced infrastructure deployment time enables. Some baseline metrics for this are provided in the IDC report.
  - Calculate the realistic and more conservative benefit expectations. Map them over the period of the business case, making allowances for ramping up to full efficiency some time after the respective workloads are migrated.

- For business productivity tools, do the following:
  - Make a realistic and a more conservative assessment of the time savings that major and a minor application upgrades could be expected to drive.
  - Estimate the average cost of people's time and effort across the impacted user base.
  - Use the figures for increased major and minor release frequency, and calculate the benefits over the term of the business case.

Because the increased developer productivity and reduced time to launch require no additional resources, add the net benefit lines for each workload into the business case cash flow model for inclusion in the discounted cashflow, NPV, ROI, MIRR, and payback calculations.

Continuous assessment and improvement

This stage of assessment focuses on two aspects:

- Ongoing detailed application assessment, for each wave of applications
- Continuous evolution and improvement of your portfolio

The first aspect, ongoing detailed application assessment, focuses on detailed discovery and analysis, down to the architecture and technology levels, to fully understand each application in a given wave, the proposed AWS design, and the migration strategy. This assessment of migration readiness is a prerequisite to starting a given migration wave.

The second aspect, continuous evolution and improvement of your portfolio, focuses on portfolio management and how you plan to improve applications over time, including the evolution and tracking of the business case.

The primary migration outcomes of this stage include the following:

- Validated migration scope for each wave
- A documented target architecture and migration strategy for applications in a given migration wave
- Identified and validated migration patterns and tooling
- Documented requirements (security, AWS infrastructure, and operations) and migration cut-over considerations for each wave

The primary optimization outcomes of this stage include the following:

- Portfolio rationalization models and business outcomes
- Proposed architecture and technology changes, and their expected benefits
- Platform requirements (security, AWS infrastructure, and operations)
- An implementation plan

Understanding continuous assessment data requirements

Data requirements for continuous assessment and improvement of the application portfolio are a combination of data requirements from previous sections. To continuously manage the portfolio migration and its evolution, see the following sections to understand data requirements:

- For wave assessment and application optimization, use the data requirements from the Prioritized applications assessment section.
- For continuous portfolio management, use the data requirements of the Portfolio analysis and migration planning section.
- For defining the wave plan, see the Wave planning section.

Detailed wave assessment

The detailed assessment of applications, ahead of a migration wave and as a key enabler for migration, has the same requirements and recommendations as the prioritized applications assessment stage. The goals are to understand in detail the current state of the applications in a given wave, and to produce a future state architecture design and migration strategy, including operational aspects, tooling, and specific migration patterns.

Apply the prioritized applications assessment to the group of applications in a given wave. Repeat this process ahead of each wave in your migration plan. The key is to schedule enough time in between the detailed assessment and the start of the wave. The amount of time needed will be dictated by the requirements of platform and migration teams that are implementing the wave requirements and performing the migrations. Work with those teams to schedule the detailed wave assessment and the wave. We recommend implementing a factory-like model emulating a production line.

Assessment for optimization and modernization

The assessment process for workload optimization and modernization already migrated into AWS is similar to the assessment of workloads to be migrated into AWS. What will change, primarily, is the sources of data to conduct the assessments. In AWS, there are several out-of-the-box tools and services that you can use to obtain more information about your applications running in AWS.

What and how to optimize and modernize your applications will be based on your unique drivers and circumstances. Optimization focuses on applying changes to the current architecture and

technology to reduce cost, adapt performance requirements, and to incorporate lessons learned. Modernization focuses on taking your application to the next level, such as adopting serverless models and microservice architectures.

Follow the guidelines of the prioritized applications assessment. To further aid your optimization and modernization efforts, see the following resources:

- AWS cost optimization provides information on IT optimization and saving on your IT costs.
- AWS Compute Optimizer recommends AWS resources for your workloads to reduce costs and improve performance by using machine learning to analyze historical utilization metrics.
- AWS cost and capacity optimization services and tools help to manage compute resources so that you can spend more time building and less time managing compute costs
- Amazon S3 Storage Lens delivers organization-wide visibility into object storage usage and activity trends. It makes actionable recommendations to improve cost-efficiency and apply data protection best practices.
- Database Freedom facilitates migration to AWS database and analytics services.
- Amazon CodeGuru is a developer tool that provides intelligent recommendations to improve code quality and identify an application's most expensive lines of code.
- AWS hybrid cloud services deliver a consistent AWS experience wherever you need it—from the cloud, to on premises, and at the edge.

Additional resources

- Cost optimization and innovation: An introduction to application modernization (blog post)
- Optimizing the cost of serverless web applications (blog post)
- Windows on AWS (blog)
- Modern applications
- Application modernization (AWS re:Invent 2020)
- AWS microservices guide

Iterating the wave plan

As the migration program moves forward and more waves are migrated, it is key to evolve the migration wave plan based on lessons learned and changing business priorities. In particular, for

long-running migration programs, it is important to reassess business drivers and organizational change, and to ensure that the migration wave plan is still valid.

Similarly, lessons learned from the migration will influence the wave plan composition and the scope of each wave. To avoid losing visibility into what is happening, keep the wave plan up to date. The plan should reflect and track what is being delivered, and it should manage and assess change to the migration scope.

Evolving and tracking the business case

As the migration proceeds, especially for long-running programs, it is inevitable that business pressures will cause migration and modernization priorities to be regularly re-examined.

We recommend that you both evolve the business case as new information becomes available, and that you track actual commercial performance against the expectations documented in the detailed business case. These recommendations include the following:

- New structural change in the organization affecting business priorities and impacting IT strategy and the application portfolio with it
- Increased commercial importance of one part of the application portfolio or the changes to it that migration and modernization are targeted to achieve
- Availability of actual resource utilization data for migrated applications, including refining sizing and quantifying and confirming cases for incremental modernization
- Availability of data on effort consumed in IT operations and support activities, and analyses of possible operational improvements and automation
- Availability of data measuring changes in software development and maintenance cycle times, software defect by development stage and service availability information, and root cause analyses for areas open to further improvement

By tracking performance against the business case, you can evolve the case to include further improvements that can be more readily assessed and quantified after migration starts. The program governance organization is much better equipped to respond to changing business pressures and to steer the transformation in a direction that drives the greatest value at a manageable and acceptable level of risk.

This is particularly important for the IT productivity, resilience, and business agility benefits within the case. These are typically both the larger and the more difficult drivers to assess ahead of time.

By tracking the performance of these drivers, the team can dive deep and resolve problems that are hindering benefits realization. Or the business case can be adjusted to prioritize initiatives that achieve the most ongoing financial performance optimization.

Resources

AWS references

- The Amazon Builders' Library
- Application modernization (AWS re:Invent 2020)
- Application portfolio assessment strategy
- AWS Architecture Center
- AWS Compute Optimizer
- AWS cost and capacity optimization services and tools
- AWS cost optimization
- Cost optimization and innovation: An introduction to application modernization (blog post)
- Discovery, Planning, and Recommendation migration tools
- AWS Documentation
- Getting Started Resource Center
- AWS Marketplace
- AWS Managed Services Partners
- AWS microservices guide
- AWS Migration Competency Partners
- Modern applications
- Optimizing the cost of serverless web applications (blog post)
- AWS Prescriptive Guidance
- AWS Professional Services
- AWS Solutions Library
- Windows on AWS (blog)

AWS services

- AWS App2Container
- AWS Application Migration Service
- Amazon CodeGuru

- AWS Control Tower
- Database Freedom
- AWS Database Migration Service
- AWS DataSync
- AWS Direct Connect
- Amazon ECS
- Amazon EKS
- AWS Fargate
- AWS Managed Services
- Migration Evaluator
- AWS Migration Hub strategy recommendations
- AWS Landing Zone
- AWS Pricing Calculator
- AWS Schema Conversion Tool
- Amazon S3 Storage Lens
- AWS Snowball Edge
- AWS Snowball Edge
- AWS VPN

Other resources

- Fostering Business and Organizational Transformation to Generate Business Value with Amazon Web Services
- IDC 2018 survey

Document history

The following table describes significant changes to this strategy. If you want to be notified about future updates, you can subscribe to an RSS feed.

Change	Description	Date
Updates	Renamed the *Portfolio discovery and initial planning* section *Discovery acceleration and initial planning*; updated the decision-tree diagram.	May 20, 2024
—	Initial publication	November 12, 2021

AWS Prescriptive Guidance glossary

The following are commonly used terms in strategies, guides, and patterns provided by AWS Prescriptive Guidance. To suggest entries, please use the **Provide feedback** link at the end of the glossary.

Numbers

7 Rs

Seven common migration strategies for moving applications to the cloud. These strategies build upon the 5 Rs that Gartner identified in 2011 and consist of the following:

- Refactor/re-architect – Move an application and modify its architecture by taking full advantage of cloud-native features to improve agility, performance, and scalability. This typically involves porting the operating system and database. Example: Migrate your on-premises Oracle database to the Amazon Aurora PostgreSQL-Compatible Edition.
- Replatform (lift and reshape) – Move an application to the cloud, and introduce some level of optimization to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Amazon Relational Database Service (Amazon RDS) for Oracle in the AWS Cloud.
- Repurchase (drop and shop) – Switch to a different product, typically by moving from a traditional license to a SaaS model. Example: Migrate your customer relationship management (CRM) system to Salesforce.com.
- Rehost (lift and shift) – Move an application to the cloud without making any changes to take advantage of cloud capabilities. Example: Migrate your on-premises Oracle database to Oracle on an EC2 instance in the AWS Cloud.
- Relocate (hypervisor-level lift and shift) – Move infrastructure to the cloud without purchasing new hardware, rewriting applications, or modifying your existing operations. You migrate servers from an on-premises platform to a cloud service for the same platform. Example: Migrate a Microsoft Hyper-V application to AWS.
- Retain (revisit) – Keep applications in your source environment. These might include applications that require major refactoring, and you want to postpone that work until a later time, and legacy applications that you want to retain, because there's no business justification for migrating them.

- Retire – Decommission or remove applications that are no longer needed in your source environment.

A

ABAC

See attribute-based access control.

abstracted services

See managed services.

ACID

See atomicity, consistency, isolation, durability.

active-active migration

A database migration method in which the source and target databases are kept in sync (by using a bidirectional replication tool or dual write operations), and both databases handle transactions from connecting applications during migration. This method supports migration in small, controlled batches instead of requiring a one-time cutover. It's more flexible but requires more work than active-passive migration.

active-passive migration

A database migration method in which in which the source and target databases are kept in sync, but only the source database handles transactions from connecting applications while data is replicated to the target database. The target database doesn't accept any transactions during migration.

aggregate function

A SQL function that operates on a group of rows and calculates a single return value for the group. Examples of aggregate functions include `SUM` and `MAX`.

AI

See artificial intelligence.

AIOps

See artificial intelligence operations.

anonymization

The process of permanently deleting personal information in a dataset. Anonymization can help protect personal privacy. Anonymized data is no longer considered to be personal data.

anti-pattern

A frequently used solution for a recurring issue where the solution is counter-productive, ineffective, or less effective than an alternative.

application control

A security approach that allows the use of only approved applications in order to help protect a system from malware.

application portfolio

A collection of detailed information about each application used by an organization, including the cost to build and maintain the application, and its business value. This information is key to the portfolio discovery and analysis process and helps identify and prioritize the applications to be migrated, modernized, and optimized.

artificial intelligence (AI)

The field of computer science that is dedicated to using computing technologies to perform cognitive functions that are typically associated with humans, such as learning, solving problems, and recognizing patterns. For more information, see What is Artificial Intelligence?

artificial intelligence operations (AIOps)

The process of using machine learning techniques to solve operational problems, reduce operational incidents and human intervention, and increase service quality. For more information about how AIOps is used in the AWS migration strategy, see the operations integration guide.

asymmetric encryption

An encryption algorithm that uses a pair of keys, a public key for encryption and a private key for decryption. You can share the public key because it isn't used for decryption, but access to the private key should be highly restricted.

atomicity, consistency, isolation, durability (ACID)

A set of software properties that guarantee the data validity and operational reliability of a database, even in the case of errors, power failures, or other problems.

attribute-based access control (ABAC)

The practice of creating fine-grained permissions based on user attributes, such as department, job role, and team name. For more information, see ABAC for AWS in the AWS Identity and Access Management (IAM) documentation.

authoritative data source

A location where you store the primary version of data, which is considered to be the most reliable source of information. You can copy data from the authoritative data source to other locations for the purposes of processing or modifying the data, such as anonymizing, redacting, or pseudonymizing it.

Availability Zone

A distinct location within an AWS Region that is insulated from failures in other Availability Zones and provides inexpensive, low-latency network connectivity to other Availability Zones in the same Region.

AWS Cloud Adoption Framework (AWS CAF)

A framework of guidelines and best practices from AWS to help organizations develop an efficient and effective plan to move successfully to the cloud. AWS CAF organizes guidance into six focus areas called perspectives: business, people, governance, platform, security, and operations. The business, people, and governance perspectives focus on business skills and processes; the platform, security, and operations perspectives focus on technical skills and processes. For example, the people perspective targets stakeholders who handle human resources (HR), staffing functions, and people management. For this perspective, AWS CAF provides guidance for people development, training, and communications to help ready the organization for successful cloud adoption. For more information, see the AWS CAF website and the AWS CAF whitepaper.

AWS Workload Qualification Framework (AWS WQF)

A tool that evaluates database migration workloads, recommends migration strategies, and provides work estimates. AWS WQF is included with AWS Schema Conversion Tool (AWS SCT). It analyzes database schemas and code objects, application code, dependencies, and performance characteristics, and provides assessment reports.

B

bad bot

A bot that is intended to disrupt or cause harm to individuals or organizations.

BCP

See business continuity planning.

behavior graph

A unified, interactive view of resource behavior and interactions over time. You can use a behavior graph with Amazon Detective to examine failed logon attempts, suspicious API calls, and similar actions. For more information, see Data in a behavior graph in the Detective documentation.

big-endian system

A system that stores the most significant byte first. See also endianness.

binary classification

A process that predicts a binary outcome (one of two possible classes). For example, your ML model might need to predict problems such as “Is this email spam or not spam?" or "Is this product a book or a car?"

bloom filter

A probabilistic, memory-efficient data structure that is used to test whether an element is a member of a set.

blue/green deployment

A deployment strategy where you create two separate but identical environments. You run the current application version in one environment (blue) and the new application version in the other environment (green). This strategy helps you quickly roll back with minimal impact.

bot

A software application that runs automated tasks over the internet and simulates human activity or interaction. Some bots are useful or beneficial, such as web crawlers that index information on the internet. Some other bots, known as *bad bots*, are intended to disrupt or cause harm to individuals or organizations.

botnet

Networks of bots that are infected by malware and are under the control of a single party, known as a *bot herder* or *bot operator*. Botnets are the best-known mechanism to scale bots and their impact.

branch

A contained area of a code repository. The first branch created in a repository is the *main branch*. You can create a new branch from an existing branch, and you can then develop features or fix bugs in the new branch. A branch you create to build a feature is commonly referred to as a *feature branch*. When the feature is ready for release, you merge the feature branch back into the main branch. For more information, see About branches (GitHub documentation).

break-glass access

In exceptional circumstances and through an approved process, a quick means for a user to gain access to an AWS account that they don't typically have permissions to access. For more information, see the Implement break-glass procedures indicator in the AWS Well-Architected guidance.

brownfield strategy

The existing infrastructure in your environment. When adopting a brownfield strategy for a system architecture, you design the architecture around the constraints of the current systems and infrastructure. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

buffer cache

The memory area where the most frequently accessed data is stored.

business capability

What a business does to generate value (for example, sales, customer service, or marketing). Microservices architectures and development decisions can be driven by business capabilities. For more information, see the Organized around business capabilities section of the Running containerized microservices on AWS whitepaper.

business continuity planning (BCP)

A plan that addresses the potential impact of a disruptive event, such as a large-scale migration, on operations and enables a business to resume operations quickly.

C

CAF

See AWS Cloud Adoption Framework.

canary deployment

The slow and incremental release of a version to end users. When you are confident, you deploy the new version and replace the current version in its entirety.

CCoE

See Cloud Center of Excellence.

CDC

See change data capture.

change data capture (CDC)

The process of tracking changes to a data source, such as a database table, and recording metadata about the change. You can use CDC for various purposes, such as auditing or replicating changes in a target system to maintain synchronization.

chaos engineering

Intentionally introducing failures or disruptive events to test a system's resilience. You can use AWS Fault Injection Service (AWS FIS) to perform experiments that stress your AWS workloads and evaluate their response.

CI/CD

See continuous integration and continuous delivery.

classification

A categorization process that helps generate predictions. ML models for classification problems predict a discrete value. Discrete values are always distinct from one another. For example, a model might need to evaluate whether or not there is a car in an image.

client-side encryption

Encryption of data locally, before the target AWS service receives it.

Cloud Center of Excellence (CCoE)

A multi-disciplinary team that drives cloud adoption efforts across an organization, including developing cloud best practices, mobilizing resources, establishing migration timelines, and leading the organization through large-scale transformations. For more information, see the CCoE posts on the AWS Cloud Enterprise Strategy Blog.

cloud computing

The cloud technology that is typically used for remote data storage and IoT device management. Cloud computing is commonly connected to edge computing technology.

cloud operating model

In an IT organization, the operating model that is used to build, mature, and optimize one or more cloud environments. For more information, see Building your Cloud Operating Model.

cloud stages of adoption

The four phases that organizations typically go through when they migrate to the AWS Cloud:

- Project – Running a few cloud-related projects for proof of concept and learning purposes
- Foundation – Making foundational investments to scale your cloud adoption (e.g., creating a landing zone, defining a CCoE, establishing an operations model)
- Migration – Migrating individual applications
- Re-invention – Optimizing products and services, and innovating in the cloud

These stages were defined by Stephen Orban in the blog post The Journey Toward Cloud-First & the Stages of Adoption on the AWS Cloud Enterprise Strategy blog. For information about how they relate to the AWS migration strategy, see the migration readiness guide.

CMDB

See configuration management database.

code repository

A location where source code and other assets, such as documentation, samples, and scripts, are stored and updated through version control processes. Common cloud repositories include GitHub or Bitbucket Cloud. Each version of the code is called a *branch*. In a microservice structure, each repository is devoted to a single piece of functionality. A single CI/CD pipeline can use multiple repositories.

cold cache

A buffer cache that is empty, not well populated, or contains stale or irrelevant data. This affects performance because the database instance must read from the main memory or disk, which is slower than reading from the buffer cache.

cold data

Data that is rarely accessed and is typically historical. When querying this kind of data, slow queries are typically acceptable. Moving this data to lower-performing and less expensive storage tiers or classes can reduce costs.

computer vision (CV)

A field of AI that uses machine learning to analyze and extract information from visual formats such as digital images and videos. For example, Amazon SageMaker AI provides image processing algorithms for CV.

configuration drift

For a workload, a configuration change from the expected state. It might cause the workload to become noncompliant, and it's typically gradual and unintentional.

configuration management database (CMDB)

A repository that stores and manages information about a database and its IT environment, including both hardware and software components and their configurations. You typically use data from a CMDB in the portfolio discovery and analysis stage of migration.

conformance pack

A collection of AWS Config rules and remediation actions that you can assemble to customize your compliance and security checks. You can deploy a conformance pack as a single entity in an AWS account and Region, or across an organization, by using a YAML template. For more information, see Conformance packs in the AWS Config documentation.

continuous integration and continuous delivery (CI/CD)

The process of automating the source, build, test, staging, and production stages of the software release process. CI/CD is commonly described as a pipeline. CI/CD can help you automate processes, improve productivity, improve code quality, and deliver faster. For more information, see Benefits of continuous delivery. CD can also stand for *continuous deployment*. For more information, see Continuous Delivery vs. Continuous Deployment.

CV

See computer vision.

D

data at rest

Data that is stationary in your network, such as data that is in storage.

data classification

A process for identifying and categorizing the data in your network based on its criticality and sensitivity. It is a critical component of any cybersecurity risk management strategy because it helps you determine the appropriate protection and retention controls for the data. Data classification is a component of the security pillar in the AWS Well-Architected Framework. For more information, see Data classification.

data drift

A meaningful variation between the production data and the data that was used to train an ML model, or a meaningful change in the input data over time. Data drift can reduce the overall quality, accuracy, and fairness in ML model predictions.

data in transit

Data that is actively moving through your network, such as between network resources.

data mesh

An architectural framework that provides distributed, decentralized data ownership with centralized management and governance.

data minimization

The principle of collecting and processing only the data that is strictly necessary. Practicing data minimization in the AWS Cloud can reduce privacy risks, costs, and your analytics carbon footprint.

data perimeter

A set of preventive guardrails in your AWS environment that help make sure that only trusted identities are accessing trusted resources from expected networks. For more information, see Building a data perimeter on AWS.

data preprocessing

To transform raw data into a format that is easily parsed by your ML model. Preprocessing data can mean removing certain columns or rows and addressing missing, inconsistent, or duplicate values.

data provenance

The process of tracking the origin and history of data throughout its lifecycle, such as how the data was generated, transmitted, and stored.

data subject

An individual whose data is being collected and processed.

data warehouse

A data management system that supports business intelligence, such as analytics. Data warehouses commonly contain large amounts of historical data, and they are typically used for queries and analysis.

database definition language (DDL)

Statements or commands for creating or modifying the structure of tables and objects in a database.

database manipulation language (DML)

Statements or commands for modifying (inserting, updating, and deleting) information in a database.

DDL

See database definition language.

deep ensemble

To combine multiple deep learning models for prediction. You can use deep ensembles to obtain a more accurate prediction or for estimating uncertainty in predictions.

deep learning

An ML subfield that uses multiple layers of artificial neural networks to identify mapping between input data and target variables of interest.

defense-in-depth

An information security approach in which a series of security mechanisms and controls are thoughtfully layered throughout a computer network to protect the confidentiality, integrity, and availability of the network and the data within. When you adopt this strategy on AWS, you add multiple controls at different layers of the AWS Organizations structure to help secure resources. For example, a defense-in-depth approach might combine multi-factor authentication, network segmentation, and encryption.

delegated administrator

In AWS Organizations, a compatible service can register an AWS member account to administer the organization's accounts and manage permissions for that service. This account is called the *delegated administrator* for that service. For more information and a list of compatible services, see Services that work with AWS Organizations in the AWS Organizations documentation.

deployment

The process of making an application, new features, or code fixes available in the target environment. Deployment involves implementing changes in a code base and then building and running that code base in the application's environments.

development environment

See environment.

detective control

A security control that is designed to detect, log, and alert after an event has occurred. These controls are a second line of defense, alerting you to security events that bypassed the preventative controls in place. For more information, see Detective controls in *Implementing security controls on AWS*.

development value stream mapping (DVSM)

A process used to identify and prioritize constraints that adversely affect speed and quality in a software development lifecycle. DVSM extends the value stream mapping process originally designed for lean manufacturing practices. It focuses on the steps and teams required to create and move value through the software development process.

digital twin

A virtual representation of a real-world system, such as a building, factory, industrial equipment, or production line. Digital twins support predictive maintenance, remote monitoring, and production optimization.

dimension table

In a star schema, a smaller table that contains data attributes about quantitative data in a fact table. Dimension table attributes are typically text fields or discrete numbers that behave like text. These attributes are commonly used for query constraining, filtering, and result set labeling.

disaster

An event that prevents a workload or system from fulfilling its business objectives in its primary deployed location. These events can be natural disasters, technical failures, or the result of human actions, such as unintentional misconfiguration or a malware attack.

disaster recovery (DR)

The strategy and process you use to minimize downtime and data loss caused by a disaster. For more information, see Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework.

DML

See database manipulation language.

domain-driven design

An approach to developing a complex software system by connecting its components to evolving domains, or core business goals, that each component serves. This concept was introduced by Eric Evans in his book, *Domain-Driven Design: Tackling Complexity in the Heart of Software* (Boston: Addison-Wesley Professional, 2003). For information about how you can use domain-driven design with the strangler fig pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

DR

See disaster recovery.

drift detection

Tracking deviations from a baselined configuration. For example, you can use AWS CloudFormation to detect drift in system resources, or you can use AWS Control Tower to detect changes in your landing zone that might affect compliance with governance requirements.

DVSM

See development value stream mapping.

E

EDA

See exploratory data analysis.

EDI

See electronic data interchange.

edge computing

The technology that increases the computing power for smart devices at the edges of an IoT network. When compared with cloud computing, edge computing can reduce communication latency and improve response time.

electronic data interchange (EDI)

The automated exchange of business documents between organizations. For more information, see What is Electronic Data Interchange.

encryption

A computing process that transforms plaintext data, which is human-readable, into ciphertext.

encryption key

A cryptographic string of randomized bits that is generated by an encryption algorithm. Keys can vary in length, and each key is designed to be unpredictable and unique.

endianness

The order in which bytes are stored in computer memory. Big-endian systems store the most significant byte first. Little-endian systems store the least significant byte first.

endpoint

See service endpoint.

endpoint service

A service that you can host in a virtual private cloud (VPC) to share with other users. You can create an endpoint service with AWS PrivateLink and grant permissions to other AWS accounts or to AWS Identity and Access Management (IAM) principals. These accounts or principals can connect to your endpoint service privately by creating interface VPC endpoints. For more

information, see Create an endpoint service in the Amazon Virtual Private Cloud (Amazon VPC) documentation.

enterprise resource planning (ERP)

A system that automates and manages key business processes (such as accounting, MES, and project management) for an enterprise.

envelope encryption

The process of encrypting an encryption key with another encryption key. For more information, see Envelope encryption in the AWS Key Management Service (AWS KMS) documentation.

environment

An instance of a running application. The following are common types of environments in cloud computing:

- development environment – An instance of a running application that is available only to the core team responsible for maintaining the application. Development environments are used to test changes before promoting them to upper environments. This type of environment is sometimes referred to as a *test environment*.
- lower environments – All development environments for an application, such as those used for initial builds and tests.
- production environment – An instance of a running application that end users can access. In a CI/CD pipeline, the production environment is the last deployment environment.
- upper environments – All environments that can be accessed by users other than the core development team. This can include a production environment, preproduction environments, and environments for user acceptance testing.

epic

In agile methodologies, functional categories that help organize and prioritize your work. Epics provide a high-level description of requirements and implementation tasks. For example, AWS CAF security epics include identity and access management, detective controls, infrastructure security, data protection, and incident response. For more information about epics in the AWS migration strategy, see the program implementation guide.

ERP

See enterprise resource planning.

exploratory data analysis (EDA)

The process of analyzing a dataset to understand its main characteristics. You collect or aggregate data and then perform initial investigations to find patterns, detect anomalies, and check assumptions. EDA is performed by calculating summary statistics and creating data visualizations.

F

fact table

The central table in a star schema. It stores quantitative data about business operations. Typically, a fact table contains two types of columns: those that contain measures and those that contain a foreign key to a dimension table.

fail fast

A philosophy that uses frequent and incremental testing to reduce the development lifecycle. It is a critical part of an agile approach.

fault isolation boundary

In the AWS Cloud, a boundary such as an Availability Zone, AWS Region, control plane, or data plane that limits the effect of a failure and helps improve the resilience of workloads. For more information, see AWS Fault Isolation Boundaries.

feature branch

See branch.

features

The input data that you use to make a prediction. For example, in a manufacturing context, features could be images that are periodically captured from the manufacturing line.

feature importance

How significant a feature is for a model's predictions. This is usually expressed as a numerical score that can be calculated through various techniques, such as Shapley Additive Explanations (SHAP) and integrated gradients. For more information, see Machine learning model interpretability with AWS.

feature transformation

To optimize data for the ML process, including enriching data with additional sources, scaling values, or extracting multiple sets of information from a single data field. This enables the ML model to benefit from the data. For example, if you break down the “2021-05-27 00:15:37” date into “2021”, “May”, “Thu”, and “15”, you can help the learning algorithm learn nuanced patterns associated with different data components.

few-shot prompting

Providing an LLM with a small number of examples that demonstrate the task and desired output before asking it to perform a similar task. This technique is an application of in-context learning, where models learn from examples (*shots*) that are embedded in prompts. Few-shot prompting can be effective for tasks that require specific formatting, reasoning, or domain knowledge. See also zero-shot prompting.

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

The use of multiple conditions to allow or deny an access request.

flash-cut migration

A database migration method that uses continuous data replication through change data capture to migrate data in the shortest time possible, instead of using a phased approach. The objective is to keep downtime to a minimum.

FM

See foundation model.

foundation model (FM)

A large deep-learning neural network that has been training on massive datasets of generalized and unlabeled data. FMs are capable of performing a wide variety of general tasks, such as understanding language, generating text and images, and conversing in natural language. For more information, see What are Foundation Models.

G

generative AI

A subset of AI models that have been trained on large amounts of data and that can use a simple text prompt to create new content and artifacts, such as images, videos, text, and audio. For more information, see What is Generative AI.

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

In Amazon CloudFront, an option to prevent users in specific countries from accessing content distributions. You can use an allow list or block list to specify approved and banned countries. For more information, see Restricting the geographic distribution of your content in the CloudFront documentation.

Gitflow workflow

An approach in which lower and upper environments use different branches in a source code repository. The Gitflow workflow is considered legacy, and the trunk-based workflow is the modern, preferred approach.

golden image

A snapshot of a system or software that is used as a template to deploy new instances of that system or software. For example, in manufacturing, a golden image can be used to provision software on multiple devices and helps improve speed, scalability, and productivity in device manufacturing operations.

greenfield strategy

The absence of existing infrastructure in a new environment. When adopting a greenfield strategy for a system architecture, you can select all new technologies without the restriction of compatibility with existing infrastructure, also known as brownfield. If you are expanding the existing infrastructure, you might blend brownfield and greenfield strategies.

guardrail

A high-level rule that helps govern resources, policies, and compliance across organizational units (OUs). *Preventive guardrails* enforce policies to ensure alignment to compliance standards. They are implemented by using service control policies and IAM permissions boundaries.

Detective guardrails detect policy violations and compliance issues, and generate alerts for remediation. They are implemented by using AWS Config, AWS Security Hub, Amazon GuardDuty, AWS Trusted Advisor, Amazon Inspector, and custom AWS Lambda checks.

H

HA

See high availability.

heterogeneous database migration

Migrating your source database to a target database that uses a different database engine (for example, Oracle to Amazon Aurora). Heterogeneous migration is typically part of a re-architecting effort, and converting the schema can be a complex task. AWS provides AWS SCT that helps with schema conversions.

high availability (HA)

The ability of a workload to operate continuously, without intervention, in the event of challenges or disasters. HA systems are designed to automatically fail over, consistently deliver high-quality performance, and handle different loads and failures with minimal performance impact.

historian modernization

An approach used to modernize and upgrade operational technology (OT) systems to better serve the needs of the manufacturing industry. A *historian* is a type of database that is used to collect and store data from various sources in a factory.

holdout data

A portion of historical, labeled data that is withheld from a dataset that is used to train a machine learning model. You can use holdout data to evaluate the model performance by comparing the model predictions against the holdout data.

homogeneous database migration

Migrating your source database to a target database that shares the same database engine (for example, Microsoft SQL Server to Amazon RDS for SQL Server). Homogeneous migration is typically part of a rehosting or replatforming effort. You can use native database utilities to migrate the schema.

hot data

Data that is frequently accessed, such as real-time data or recent translational data. This data typically requires a high-performance storage tier or class to provide fast query responses.

hotfix

An urgent fix for a critical issue in a production environment. Due to its urgency, a hotfix is usually made outside of the typical DevOps release workflow.

hypercare period

Immediately following cutover, the period of time when a migration team manages and monitors the migrated applications in the cloud in order to address any issues. Typically, this period is 1–4 days in length. At the end of the hypercare period, the migration team typically transfers responsibility for the applications to the cloud operations team.

I

IaC

See infrastructure as code.

identity-based policy

A policy attached to one or more IAM principals that defines their permissions within the AWS Cloud environment.

idle application

An application that has an average CPU and memory usage between 5 and 20 percent over a period of 90 days. In a migration project, it is common to retire these applications or retain them on premises.

IIoT

See industrial Internet of Things.

immutable infrastructure

A model that deploys new infrastructure for production workloads instead of updating, patching, or modifying the existing infrastructure. Immutable infrastructures are inherently more consistent, reliable, and predictable than mutable infrastructure. For more information, see the Deploy using immutable infrastructure best practice in the AWS Well-Architected Framework.

inbound (ingress) VPC

In an AWS multi-account architecture, a VPC that accepts, inspects, and routes network connections from outside an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

incremental migration

A cutover strategy in which you migrate your application in small parts instead of performing a single, full cutover. For example, you might move only a few microservices or users to the new system initially. After you verify that everything is working properly, you can incrementally move additional microservices or users until you can decommission your legacy system. This strategy reduces the risks associated with large migrations.

Industry 4.0

A term that was introduced by Klaus Schwab in 2016 to refer to the modernization of manufacturing processes through advances in connectivity, real-time data, automation, analytics, and AI/ML.

infrastructure

All of the resources and assets contained within an application's environment.

infrastructure as code (IaC)

The process of provisioning and managing an application's infrastructure through a set of configuration files. IaC is designed to help you centralize infrastructure management, standardize resources, and scale quickly so that new environments are repeatable, reliable, and consistent.

industrial Internet of Things (IIoT)

The use of internet-connected sensors and devices in the industrial sectors, such as manufacturing, energy, automotive, healthcare, life sciences, and agriculture. For more information, see Building an industrial Internet of Things (IIoT) digital transformation strategy.

inspection VPC

In an AWS multi-account architecture, a centralized VPC that manages inspections of network traffic between VPCs (in the same or different AWS Regions), the internet, and on-premises networks. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

Internet of Things (IoT)

The network of connected physical objects with embedded sensors or processors that communicate with other devices and systems through the internet or over a local communication network. For more information, see What is IoT?

interpretability

A characteristic of a machine learning model that describes the degree to which a human can understand how the model's predictions depend on its inputs. For more information, see Machine learning model interpretability with AWS.

IoT

See Internet of Things.

IT information library (ITIL)

A set of best practices for delivering IT services and aligning these services with business requirements. ITIL provides the foundation for ITSM.

IT service management (ITSM)

Activities associated with designing, implementing, managing, and supporting IT services for an organization. For information about integrating cloud operations with ITSM tools, see the operations integration guide.

ITIL

See IT information library.

ITSM

See IT service management.

L

label-based access control (LBAC)

An implementation of mandatory access control (MAC) where the users and the data itself are each explicitly assigned a security label value. The intersection between the user security label and data security label determines which rows and columns can be seen by the user.

landing zone

A landing zone is a well-architected, multi-account AWS environment that is scalable and secure. This is a starting point from which your organizations can quickly launch and deploy workloads and applications with confidence in their security and infrastructure environment. For more information about landing zones, see Setting up a secure and scalable multi-account AWS environment.

large language model (LLM)

A deep learning AI model that is pretrained on a vast amount of data. An LLM can perform multiple tasks, such as answering questions, summarizing documents, translating text into other languages, and completing sentences. For more information, see What are LLMs.

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

The security best practice of granting the minimum permissions required to perform a task. For more information, see Apply least-privilege permissions in the IAM documentation.

lift and shift

See 7 Rs.

little-endian system

A system that stores the least significant byte first. See also endianness.

LLM

See large language model.

lower environments

See environment.

M

machine learning (ML)

A type of artificial intelligence that uses algorithms and techniques for pattern recognition and learning. ML analyzes and learns from recorded data, such as Internet of Things (IoT) data, to generate a statistical model based on patterns. For more information, see Machine Learning.

main branch

See branch.

malware

Software that is designed to compromise computer security or privacy. Malware might disrupt computer systems, leak sensitive information, or gain unauthorized access. Examples of malware include viruses, worms, ransomware, Trojan horses, spyware, and keyloggers.

managed services

AWS services for which AWS operates the infrastructure layer, the operating system, and platforms, and you access the endpoints to store and retrieve data. Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB are examples of managed services. These are also known as *abstracted services*.

manufacturing execution system (MES)

A software system for tracking, monitoring, documenting, and controlling production processes that convert raw materials to finished products on the shop floor.

MAP

See Migration Acceleration Program.

mechanism

A complete process in which you create a tool, drive adoption of the tool, and then inspect the results in order to make adjustments. A mechanism is a cycle that reinforces and improves itself as it operates. For more information, see Building mechanisms in the AWS Well-Architected Framework.

member account

All AWS accounts other than the management account that are part of an organization in AWS Organizations. An account can be a member of only one organization at a time.

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

A lightweight, machine-to-machine (M2M) communication protocol, based on the publish/subscribe pattern, for resource-constrained IoT devices.

microservice

A small, independent service that communicates over well-defined APIs and is typically owned by small, self-contained teams. For example, an insurance system might include microservices that map to business capabilities, such as sales or marketing, or subdomains, such as purchasing, claims, or analytics. The benefits of microservices include agility, flexible scaling, easy deployment, reusable code, and resilience. For more information, see Integrating microservices by using AWS serverless services.

microservices architecture

An approach to building an application with independent components that run each application process as a microservice. These microservices communicate through a well-defined interface by using lightweight APIs. Each microservice in this architecture can be updated, deployed, and scaled to meet demand for specific functions of an application. For more information, see Implementing microservices on AWS.

Migration Acceleration Program (MAP)

An AWS program that provides consulting support, training, and services to help organizations build a strong operational foundation for moving to the cloud, and to help offset the initial cost of migrations. MAP includes a migration methodology for executing legacy migrations in a methodical way and a set of tools to automate and accelerate common migration scenarios.

migration at scale

The process of moving the majority of the application portfolio to the cloud in waves, with more applications moved at a faster rate in each wave. This phase uses the best practices and lessons learned from the earlier phases to implement a *migration factory* of teams, tools, and processes to streamline the migration of workloads through automation and agile delivery. This is the third phase of the AWS migration strategy.

migration factory

Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners,

migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the discussion of migration factories and the Cloud Migration Factory guide in this content set.

migration metadata

The information about the application and server that is needed to complete the migration. Each migration pattern requires a different set of migration metadata. Examples of migration metadata include the target subnet, security group, and AWS account.

migration pattern

A repeatable migration task that details the migration strategy, the migration destination, and the migration application or service used. Example: Rehost migration to Amazon EC2 with AWS Application Migration Service.

Migration Portfolio Assessment (MPA)

An online tool that provides information for validating the business case for migrating to the AWS Cloud. MPA provides detailed portfolio assessment (server right-sizing, pricing, TCO comparisons, migration cost analysis) as well as migration planning (application data analysis and data collection, application grouping, migration prioritization, and wave planning). The MPA tool (requires login) is available free of charge to all AWS consultants and APN Partner consultants.

Migration Readiness Assessment (MRA)

The process of gaining insights about an organization's cloud readiness status, identifying strengths and weaknesses, and building an action plan to close identified gaps, using the AWS CAF. For more information, see the migration readiness guide. MRA is the first phase of the AWS migration strategy.

migration strategy

The approach used to migrate a workload to the AWS Cloud. For more information, see the 7 Rs entry in this glossary and see Mobilize your organization to accelerate large-scale migrations.

ML

See machine learning.

modernization

Transforming an outdated (legacy or monolithic) application and its infrastructure into an agile, elastic, and highly available system in the cloud to reduce costs, gain efficiencies, and take advantage of innovations. For more information, see Strategy for modernizing applications in the AWS Cloud.

modernization readiness assessment

An evaluation that helps determine the modernization readiness of an organization's applications; identifies benefits, risks, and dependencies; and determines how well the organization can support the future state of those applications. The outcome of the assessment is a blueprint of the target architecture, a roadmap that details development phases and milestones for the modernization process, and an action plan for addressing identified gaps. For more information, see Evaluating modernization readiness for applications in the AWS Cloud.

monolithic applications (monoliths)

Applications that run as a single service with tightly coupled processes. Monolithic applications have several drawbacks. If one application feature experiences a spike in demand, the entire architecture must be scaled. Adding or improving a monolithic application's features also becomes more complex when the code base grows. To address these issues, you can use a microservices architecture. For more information, see Decomposing monoliths into microservices.

MPA

See Migration Portfolio Assessment.

MQTT

See Message Queuing Telemetry Transport.

multiclass classification

A process that helps generate predictions for multiple classes (predicting one of more than two outcomes). For example, an ML model might ask "Is this product a book, car, or phone?" or "Which product category is most interesting to this customer?"

mutable infrastructure

A model that updates and modifies the existing infrastructure for production workloads. For improved consistency, reliability, and predictability, the AWS Well-Architected Framework recommends the use of immutable infrastructure as a best practice.

O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

A migration method in which the source workload is taken down during the migration process. This method involves extended downtime and is typically used for small, non-critical workloads.

OI

See operations integration.

OLA

See operational-level agreement.

online migration

A migration method in which the source workload is copied to the target system without being taken offline. Applications that are connected to the workload can continue to function during the migration. This method involves zero to minimal downtime and is typically used for critical production workloads.

OPC-UA

See Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

A machine-to-machine (M2M) communication protocol for industrial automation. OPC-UA provides an interoperability standard with data encryption, authentication, and authorization schemes.

operational-level agreement (OLA)

An agreement that clarifies what functional IT groups promise to deliver to each other, to support a service-level agreement (SLA).

operational readiness review (ORR)

A checklist of questions and associated best practices that help you understand, evaluate, prevent, or reduce the scope of incidents and possible failures. For more information, see Operational Readiness Reviews (ORR) in the AWS Well-Architected Framework.

operational technology (OT)

Hardware and software systems that work with the physical environment to control industrial operations, equipment, and infrastructure. In manufacturing, the integration of OT and information technology (IT) systems is a key focus for Industry 4.0 transformations.

operations integration (OI)

The process of modernizing operations in the cloud, which involves readiness planning, automation, and integration. For more information, see the operations integration guide.

organization trail

A trail that's created by AWS CloudTrail that logs all events for all AWS accounts in an organization in AWS Organizations. This trail is created in each AWS account that's part of the organization and tracks the activity in each account. For more information, see Creating a trail for an organization in the CloudTrail documentation.

organizational change management (OCM)

A framework for managing major, disruptive business transformations from a people, culture, and leadership perspective. OCM helps organizations prepare for, and transition to, new systems and strategies by accelerating change adoption, addressing transitional issues, and driving cultural and organizational changes. In the AWS migration strategy, this framework is called *people acceleration*, because of the speed of change required in cloud adoption projects. For more information, see the OCM guide.

origin access control (OAC)

In CloudFront, an enhanced option for restricting access to secure your Amazon Simple Storage Service (Amazon S3) content. OAC supports all S3 buckets in all AWS Regions, server-side encryption with AWS KMS (SSE-KMS), and dynamic `PUT` and `DELETE` requests to the S3 bucket.

origin access identity (OAI)

In CloudFront, an option for restricting access to secure your Amazon S3 content. When you use OAI, CloudFront creates a principal that Amazon S3 can authenticate with. Authenticated principals can access content in an S3 bucket only through a specific CloudFront distribution. See also OAC, which provides more granular and enhanced access control.

ORR

See operational readiness review.

OT

See operational technology.

outbound (egress) VPC

In an AWS multi-account architecture, a VPC that handles network connections that are initiated from within an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

P

permissions boundary

An IAM management policy that is attached to IAM principals to set the maximum permissions that the user or role can have. For more information, see Permissions boundaries in the IAM documentation.

personally identifiable information (PII)

Information that, when viewed directly or paired with other related data, can be used to reasonably infer the identity of an individual. Examples of PII include names, addresses, and contact information.

PII

See personally identifiable information.

playbook

A set of predefined steps that capture the work associated with migrations, such as delivering core operations functions in the cloud. A playbook can take the form of scripts, automated runbooks, or a summary of processes or steps required to operate your modernized environment.

PLC

See programmable logic controller.

PLM

See product lifecycle management.

policy

An object that can define permissions (see identity-based policy), specify access conditions (see resource-based policy), or define the maximum permissions for all accounts in an organization in AWS Organizations (see service control policy).

polyglot persistence

Independently choosing a microservice's data storage technology based on data access patterns and other requirements. If your microservices have the same data storage technology, they can encounter implementation challenges or experience poor performance. Microservices are more easily implemented and achieve better performance and scalability if they use the data store best adapted to their requirements. For more information, see Enabling data persistence in microservices.

portfolio assessment

A process of discovering, analyzing, and prioritizing the application portfolio in order to plan the migration. For more information, see Evaluating migration readiness.

predicate

A query condition that returns `true` or `false`, commonly located in a `WHERE` clause.

predicate pushdown

A database query optimization technique that filters the data in the query before transfer. This reduces the amount of data that must be retrieved and processed from the relational database, and it improves query performance.

preventative control

A security control that is designed to prevent an event from occurring. These controls are a first line of defense to help prevent unauthorized access or unwanted changes to your network. For more information, see Preventative controls in *Implementing security controls on AWS*.

principal

An entity in AWS that can perform actions and access resources. This entity is typically a root user for an AWS account, an IAM role, or a user. For more information, see *Principal* in Roles terms and concepts in the IAM documentation.

privacy by design

A system engineering approach that takes privacy into account through the whole development process.

private hosted zones

A container that holds information about how you want Amazon Route 53 to respond to DNS queries for a domain and its subdomains within one or more VPCs. For more information, see Working with private hosted zones in the Route 53 documentation.

proactive control

A security control designed to prevent the deployment of noncompliant resources. These controls scan resources before they are provisioned. If the resource is not compliant with the control, then it isn't provisioned. For more information, see the Controls reference guide in the AWS Control Tower documentation and see Proactive controls in *Implementing security controls on AWS*.

product lifecycle management (PLM)

The management of data and processes for a product throughout its entire lifecycle, from design, development, and launch, through growth and maturity, to decline and removal.

production environment

See environment.

programmable logic controller (PLC)

In manufacturing, a highly reliable, adaptable computer that monitors machines and automates manufacturing processes.

prompt chaining

Using the output of one LLM prompt as the input for the next prompt to generate better responses. This technique is used to break down a complex task into subtasks, or to iteratively refine or expand a preliminary response. It helps improve the accuracy and relevance of a model's responses and allows for more granular, personalized results.

pseudonymization

The process of replacing personal identifiers in a dataset with placeholder values. Pseudonymization can help protect personal privacy. Pseudonymized data is still considered to be personal data.

publish/subscribe (pub/sub)

A pattern that enables asynchronous communications among microservices to improve scalability and responsiveness. For example, in a microservices-based MES, a microservice can publish event messages to a channel that other microservices can subscribe to. The system can add new microservices without changing the publishing service.

Q

query plan

A series of steps, like instructions, that are used to access the data in a SQL relational database system.

query plan regression

When a database service optimizer chooses a less optimal plan than it did before a given change to the database environment. This can be caused by changes to statistics, constraints, environment settings, query parameter bindings, and updates to the database engine.

R

RACI matrix

See responsible, accountable, consulted, informed (RACI).

RAG

See Retrieval Augmented Generation.

ransomware

A malicious software that is designed to block access to a computer system or data until a payment is made.

RASCI matrix

See responsible, accountable, consulted, informed (RACI).

RCAC

See row and column access control.

read replica

A copy of a database that's used for read-only purposes. You can route queries to the read replica to reduce the load on your primary database.

re-architect

See 7 Rs.

recovery point objective (RPO)

The maximum acceptable amount of time since the last data recovery point. This determines what is considered an acceptable loss of data between the last recovery point and the interruption of service.

recovery time objective (RTO)

The maximum acceptable delay between the interruption of service and restoration of service.

refactor

See 7 Rs.

Region

A collection of AWS resources in a geographic area. Each AWS Region is isolated and independent of the others to provide fault tolerance, stability, and resilience. For more information, see Specify which AWS Regions your account can use.

regression

An ML technique that predicts a numeric value. For example, to solve the problem of "What price will this house sell for?" an ML model could use a linear regression model to predict a house's sale price based on known facts about the house (for example, the square footage).

rehost

See 7 Rs.

release

In a deployment process, the act of promoting changes to a production environment.

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resiliency

An application's ability to resist or recover from disruptions. High availability and disaster recovery are common considerations when planning for resiliency in the AWS Cloud. For more information, see AWS Cloud Resilience.

resource-based policy

A policy attached to a resource, such as an Amazon S3 bucket, an endpoint, or an encryption key. This type of policy specifies which principals are allowed access, supported actions, and any other conditions that must be met.

responsible, accountable, consulted, informed (RACI) matrix

A matrix that defines the roles and responsibilities for all parties involved in migration activities and cloud operations. The matrix name is derived from the responsibility types defined in the matrix: responsible (R), accountable (A), consulted (C), and informed (I). The support (S) type is optional. If you include support, the matrix is called a *RASCI matrix*, and if you exclude it, it's called a *RACI matrix*.

responsive control

A security control that is designed to drive remediation of adverse events or deviations from your security baseline. For more information, see Responsive controls in *Implementing security controls on AWS*.

retain

See 7 Rs.

retire

See 7 Rs.

Retrieval Augmented Generation (RAG)

A generative AI technology in which an LLM references an authoritative data source that is outside of its training data sources before generating a response. For example, a RAG model might perform a semantic search of an organization's knowledge base or custom data. For more information, see What is RAG.

rotation

The process of periodically updating a secret to make it more difficult for an attacker to access the credentials.

row and column access control (RCAC)

The use of basic, flexible SQL expressions that have defined access rules. RCAC consists of row permissions and column masks.

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

A set of manual or automated procedures required to perform a specific task. These are typically built to streamline repetitive operations or procedures with high error rates.

S

SAML 2.0

An open standard that many identity providers (IdPs) use. This feature enables federated single sign-on (SSO), so users can log into the AWS Management Console or call the AWS API operations without you having to create user in IAM for everyone in your organization. For more information about SAML 2.0-based federation, see About SAML 2.0-based federation in the IAM documentation.

SCADA

See supervisory control and data acquisition.

SCP

See service control policy.

secret

In AWS Secrets Manager, confidential or restricted information, such as a password or user credentials, that you store in encrypted form. It consists of the secret value and its metadata.

The secret value can be binary, a single string, or multiple strings. For more information, see What's in a Secrets Manager secret? in the Secrets Manager documentation.

security by design

A system engineering approach that takes security into account through the whole development process.

security control

A technical or administrative guardrail that prevents, detects, or reduces the ability of a threat actor to exploit a security vulnerability. There are four primary types of security controls: preventative, detective, responsive, and proactive.

security hardening

The process of reducing the attack surface to make it more resistant to attacks. This can include actions such as removing resources that are no longer needed, implementing the security best practice of granting least privilege, or deactivating unnecessary features in configuration files.

security information and event management (SIEM) system

Tools and services that combine security information management (SIM) and security event management (SEM) systems. A SIEM system collects, monitors, and analyzes data from servers, networks, devices, and other sources to detect threats and security breaches, and to generate alerts.

security response automation

A predefined and programmed action that is designed to automatically respond to or remediate a security event. These automations serve as detective or responsive security controls that help you implement AWS security best practices. Examples of automated response actions include modifying a VPC security group, patching an Amazon EC2 instance, or rotating credentials.

server-side encryption

Encryption of data at its destination, by the AWS service that receives it.

service control policy (SCP)

A policy that provides centralized control over permissions for all accounts in an organization in AWS Organizations. SCPs define guardrails or set limits on actions that an administrator can delegate to users or roles. You can use SCPs as allow lists or deny lists, to specify which services or actions are permitted or prohibited. For more information, see Service control policies in the AWS Organizations documentation.

service endpoint

The URL of the entry point for an AWS service. You can use the endpoint to connect programmatically to the target service. For more information, see AWS service endpoints in *AWS General Reference*.

service-level agreement (SLA)

An agreement that clarifies what an IT team promises to deliver to their customers, such as service uptime and performance.

service-level indicator (SLI)

A measurement of a performance aspect of a service, such as its error rate, availability, or throughput.

service-level objective (SLO)

A target metric that represents the health of a service, as measured by a service-level indicator.

shared responsibility model

A model describing the responsibility you share with AWS for cloud security and compliance. AWS is responsible for security *of* the cloud, whereas you are responsible for security *in* the cloud. For more information, see Shared responsibility model.

SIEM

See security information and event management system.

single point of failure (SPOF)

A failure in a single, critical component of an application that can disrupt the system.

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

A pattern for scaling and accelerating modernization projects. As new features and product releases are defined, the core team splits up to create new product teams. This helps scale your

organization's capabilities and services, improves developer productivity, and supports rapid innovation. For more information, see Phased approach to modernizing applications in the AWS Cloud.

SPOF

See single point of failure.

star schema

A database organizational structure that uses one large fact table to store transactional or measured data and uses one or more smaller dimensional tables to store data attributes. This structure is designed for use in a data warehouse or for business intelligence purposes.

strangler fig pattern

An approach to modernizing monolithic systems by incrementally rewriting and replacing system functionality until the legacy system can be decommissioned. This pattern uses the analogy of a fig vine that grows into an established tree and eventually overcomes and replaces its host. The pattern was introduced by Martin Fowler as a way to manage risk when rewriting monolithic systems. For an example of how to apply this pattern, see Modernizing legacy Microsoft ASP.NET (ASMX) web services incrementally by using containers and Amazon API Gateway.

subnet

A range of IP addresses in your VPC. A subnet must reside in a single Availability Zone.

supervisory control and data acquisition (SCADA)

In manufacturing, a system that uses hardware and software to monitor physical assets and production operations.

symmetric encryption

An encryption algorithm that uses the same key to encrypt and decrypt the data.

synthetic testing

Testing a system in a way that simulates user interactions to detect potential issues or to monitor performance. You can use Amazon CloudWatch Synthetics to create these tests.

system prompt

A technique for providing context, instructions, or guidelines to an LLM to direct its behavior. System prompts help set context and establish rules for interactions with users.

T

tags

Key-value pairs that act as metadata for organizing your AWS resources. Tags can help you manage, identify, organize, search for, and filter resources. For more information, see Tagging your AWS resources.

target variable

The value that you are trying to predict in supervised ML. This is also referred to as an *outcome variable*. For example, in a manufacturing setting the target variable could be a product defect.

task list

A tool that is used to track progress through a runbook. A task list contains an overview of the runbook and a list of general tasks to be completed. For each general task, it includes the estimated amount of time required, the owner, and the progress.

test environment

See environment.

training

To provide data for your ML model to learn from. The training data must contain the correct answer. The learning algorithm finds patterns in the training data that map the input data attributes to the target (the answer that you want to predict). It outputs an ML model that captures these patterns. You can then use the ML model to make predictions on new data for which you don't know the target.

transit gateway

A network transit hub that you can use to interconnect your VPCs and on-premises networks. For more information, see What is a transit gateway in the AWS Transit Gateway documentation.

trunk-based workflow

An approach in which developers build and test features locally in a feature branch and then merge those changes into the main branch. The main branch is then built to the development, preproduction, and production environments, sequentially.

trusted access

Granting permissions to a service that you specify to perform tasks in your organization in AWS Organizations and in its accounts on your behalf. The trusted service creates a service-linked role in each account, when that role is needed, to perform management tasks for you. For more information, see Using AWS Organizations with other AWS services in the AWS Organizations documentation.

tuning

To change aspects of your training process to improve the ML model's accuracy. For example, you can train the ML model by generating a labeling set, adding labels, and then repeating these steps several times under different settings to optimize the model.

two-pizza team

A small DevOps team that you can feed with two pizzas. A two-pizza team size ensures the best possible opportunity for collaboration in software development.

U

uncertainty

A concept that refers to imprecise, incomplete, or unknown information that can undermine the reliability of predictive ML models. There are two types of uncertainty: *Epistemic uncertainty* is caused by limited, incomplete data, whereas *aleatoric uncertainty* is caused by the noise and randomness inherent in the data. For more information, see the Quantifying uncertainty in deep learning systems guide.

undifferentiated tasks

Also known as *heavy lifting*, work that is necessary to create and operate an application but that doesn't provide direct value to the end user or provide competitive advantage. Examples of undifferentiated tasks include procurement, maintenance, and capacity planning.

upper environments

See environment.

V

vacuuming

A database maintenance operation that involves cleaning up after incremental updates to reclaim storage and improve performance.

version control

Processes and tools that track changes, such as changes to source code in a repository.

VPC peering

A connection between two VPCs that allows you to route traffic by using private IP addresses. For more information, see What is VPC peering in the Amazon VPC documentation.

vulnerability

A software or hardware flaw that compromises the security of the system.

W

warm cache

A buffer cache that contains current, relevant data that is frequently accessed. The database instance can read from the buffer cache, which is faster than reading from the main memory or disk.

warm data

Data that is infrequently accessed. When querying this kind of data, moderately slow queries are typically acceptable.

window function

A SQL function that performs a calculation on a group of rows that relate in some way to the current record. Window functions are useful for processing tasks, such as calculating a moving average or accessing the value of rows based on the relative position of the current row.

workload

A collection of resources and code that delivers business value, such as a customer-facing application or backend process.

workstream

Functional groups in a migration project that are responsible for a specific set of tasks. Each workstream is independent but supports the other workstreams in the project. For example, the portfolio workstream is responsible for prioritizing applications, wave planning, and collecting migration metadata. The portfolio workstream delivers these assets to the migration workstream, which then migrates the servers and applications.

WORM

See write once, read many.

WQF

See AWS Workload Qualification Framework.

write once, read many (WORM)

A storage model that writes data a single time and prevents the data from being deleted or modified. Authorized users can read the data as many times as needed, but they cannot change it. This data storage infrastructure is considered immutable.

Z

zero-day exploit

An attack, typically malware, that takes advantage of a zero-day vulnerability.

zero-day vulnerability

An unmitigated flaw or vulnerability in a production system. Threat actors can use this type of vulnerability to attack the system. Developers frequently become aware of the vulnerability as a result of the attack.

zero-shot prompting

Providing an LLM with instructions for performing a task but no examples (*shots*) that can help guide it. The LLM must use its pre-trained knowledge to handle the task. The effectiveness of zero-shot prompting depends on the complexity of the task and the quality of the prompt. See also few-shot prompting.

zombie application

An application that has an average CPU and memory usage below 5 percent. In a migration project, it is common to retire these applications.