Strategy and best practices for AWS large migrations

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Strategy and best practices for AWS large migrations

Table of Contents

Strategy and best practices for AWS large migrations

Amazon Web Services (contributors)

May 2022 (document history)

Many AWS customers want to migrate a large number of servers and applications to the AWS Cloud as fast as possible with the least impact to their business. Your organization might be starting a large migration project because a data center lease is approaching renewal or termination or because your organization is taking the first steps in a technology transformation. However, large scale is not quantified only by the number of servers in scope. It also accounts for the level of organizational transformation that results from the migrations, considering complexities such as people, processes, technology, and priorities.

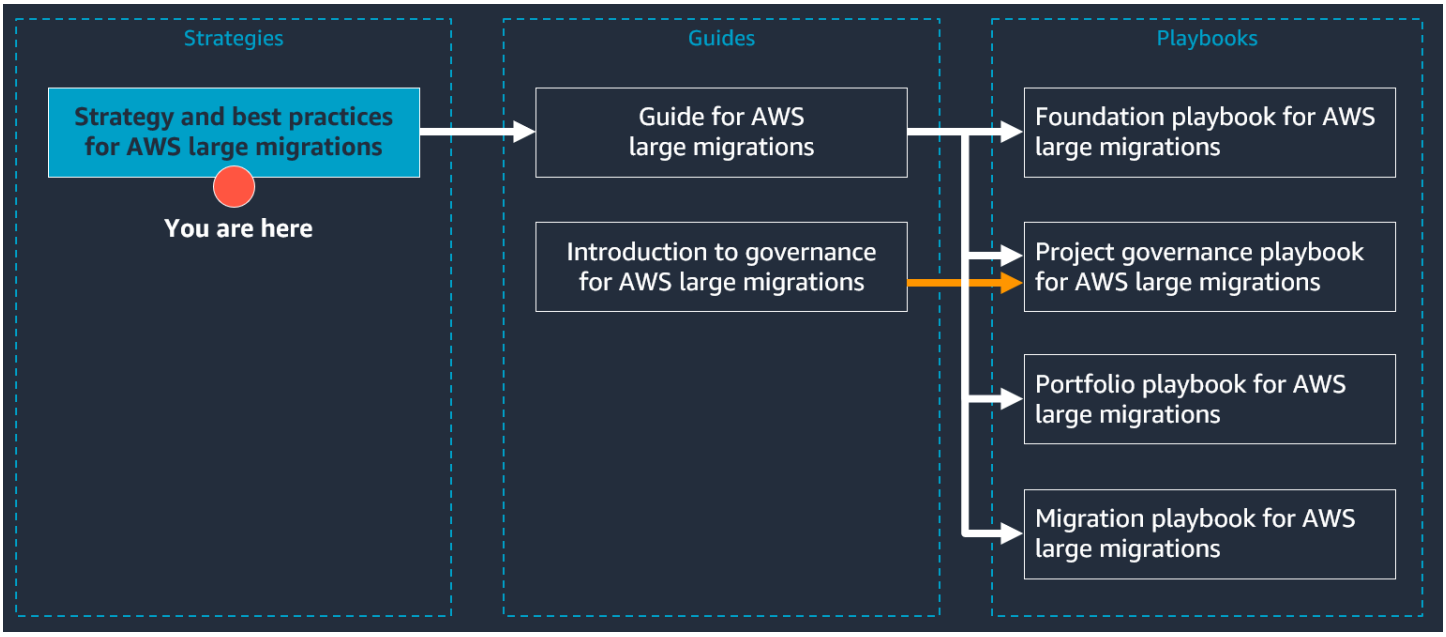
This guide focuses on your ability to move at scale to AWS. You can migrate existing applications with little to no change. You can use the cloud as a launch point to take those applications to cloud-native or serverless technologies, and you can modernize the applications to unlock additional business benefits.

This guide discusses best practices for large-scale migrations and provides use cases from customers across various segments, such as financial services, and healthcare. It also provides real-world examples of lessons learned during customer migrations to AWS. The aim of this guide is to assist customers who are at the initial stages of a large-scale migration. However, the best practices and strategies in this guide can be beneficial at any stage of the migration journey. It's assumed that you already have a 100-level knowledge of AWS services and that you're aware of the AWS recommended process for migrating.

Guidance for large migrations

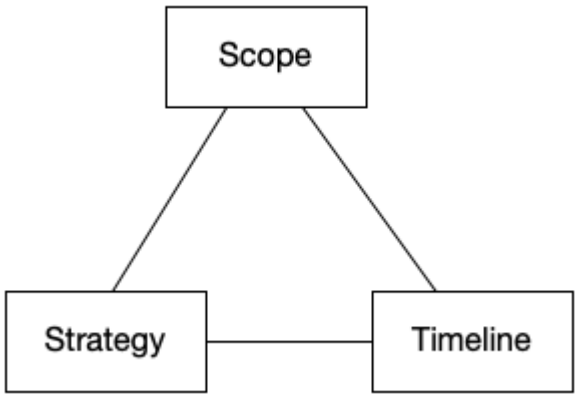
Migrating 300 or more servers is considered a large migration. The people, process, and technology challenges of a large migration project are typically new to most enterprises. This document is part of an AWS Prescriptive Guidance series about large migrations to the AWS Cloud. This series is designed to help you apply the correct strategy and best practices from the outset, to streamline your journey to the cloud.

The following figure shows the other documents in this series. Review the strategy first, then the guides, and then proceed to the playbooks. To access the complete series, see Large migrations to the AWS Cloud.


Strategies
Strategy and best practices for AWS large migrations
You are here
Guides
Guide for AWS large migrations
Introduction to governance for AWS large migrations
Playbooks
Foundation playbook for AWS large migrations
Project governance playbook for AWS large migrations
Portfolio playbook for AWS large migrations
Migration playbook for AWS large migrations

Scope, strategy, and timeline

Three key elements make up the building blocks of all programs and their relevance in large migrations: scope, strategy, and the timeline.



To set the stage for your migration journey, these elements must be aligned and understood from the start of a migration program. Any changes to one of these elements will affect the others. Realignment must be factored into every change, no matter how basic or sensible the change might seem.

Scope – What are you migrating?

It's common for the total scope of the program to be undefined, even when you're half way through the migration. This is because various factors might not be unpacked until the later stages. For example, halfway through your migration, you might uncover a pocket of shadow IT that was not recorded in your configuration management database (CMDB). Alternatively, the planning might have focused on a server view without considering the supporting network and security services that are required for those applications to run (such as VPN connections to AWS Partners, and certificate authorities to sign certificates). We recommend investing some time in defining the scope, working backwards from your target business outcome. You might end up using discovery tooling to uncover assets, a best practice that will be discussed later in this guide.

The scope will change, because large migrations come with unknowns. These unknowns could be in the form of systems that have become part of the archeology of the environment with little to no understanding of their relevance, or production incidents that cause delays and shifts to the plans you have made. The key is to be flexible and have contingency plans in place to keep the program moving forward.

Strategy – Why do you want to migrate?

You might be planning to migrate to AWS for one or more of the following reasons:

- Your application teams want to implement new CI/CD pipelines, deploy the latest application stacks, or modernize legacy platforms that are out of support.
- Your infrastructure team must get out of an aging data center quickly before the lease expires and the provider turns the power off.
- The board has decided that you need to move to the cloud as a strategic direction, allowing for a fast pace of change in the business's future.

Whatever the reason, all these reasons and more will be on the minds of your business and IT organizations. It's key to understand what your drivers are, to communicate them, and to prioritize them. Each additional driver potentially adds time, costs, scope, and risks to your already-large migration. Being fully aware of the impact that the strategy has on the timeline and scope is key.

After you define your migration strategy, one of the main keys to success is alignment of requirements across the various stakeholders and teams. Performing the migration requires different teams across the organization, including Infrastructure, Security, Application, and Operations. These teams will have individual priorities and other projects that might have already commenced. If these teams are working toward different timelines and priorities, it's more challenging to agree on and implement a migration plan. The migration team and key stakeholders must ensure that all involved teams work toward a single goal and align their priorities with a single timeline of migrations.

We recommend exploring how the desired business outcomes can be aligned across the various teams. For example, migrating to AWS and using AWS Key Management Service (AWS KMS) to encrypt storage at rest might satisfy both the migration and security goals.

Frequently, businesses want to modernize applications, which can result in infrastructure upgrades, while the infrastructure team wants to be frugal and minimize infrastructure changes. The mindset for large migrations should be as basic as possible. The teams involved must avoid trying to do everything at once.

To achieve this, set the right expectations early in the project. The key message should be "Migrate first, then modernize." This approach not only enables organizations to reduce technical debt and operate at scale eventually, it opens avenues for different modernization approaches by using the scalability and agility that the AWS Cloud can provide. Thinking long term will help infrastructure

teams to streamline infrastructure deployment and management. As a result the business can have faster feature release cycles.

Timeline – When do you need to complete the migration?

Depending on your business case, you must ensure you are not taking on more than is possible to achieve in the time allocated. If your driver for migrating is based on a fixed date of completion, you must choose the strategy that meets that timeline requirement. Most large migrations are based on these time-based constraints, so the migration strategies must have defined, fixed timelines and outcomes, with little room for extensions or overrun.

In these time-sensitive types of migrations, we recommend the “Migrate first, then modernize” approach. This helps set expectations and encourages the teams to ensure that their individual project plans and budgets are aligned with the overall migration goal. It’s important to find out any disagreements as early as possible in the project, fail fast and address the disagreements at the Steering Committee level, and engage the right stakeholders to ensure that alignment is in place.

Conversely, if your main goal of migration is to gain the benefits of application modernization, this must be called out early in the program. Many programs start with an initial goal based on a fixed deadline, and they don’t plan for the requirements from stakeholders who want to resolve outstanding issues and problems. In some cases, these issues have been present for years in the source systems, but now they become artificial blockers to migration.

Modernization activities during a migration can affect the functionality of business applications. Even what is perceived to be a small upgrade, such as an operating system version change, can have a major effect on the program timelines. These should not be considered trivial.

Best practices for large migrations

Large migrations can become challenging, depending on factors that govern how an organization functions. This section covers some of the key factors that can simplify large migrations if addressed during the initial phases of the effort and tracked throughout the project.

The following best practices for large migrations are based on data captured from other customers. The best practices are divided into three categories:

- People
- Technology
- Processes

People perspective

This section focuses on the following key areas of the people perspective:

- Executive support – Identifying a single-threaded leader who's empowered to make decisions
- Team collaboration and ownership – Collaborating among various teams
- Training – Proactively training teams on the various tooling

Executive support

In this section:

Identify a single-threaded leader

When starting a large migration, it's important to identify a single-threaded technical leader who is 100 percent dedicated to the project and accountable. That leader is empowered to make decisions, help avoid silos, and streamline work-streams by maintaining consistent priorities.

A large migration global customer was able to scale from one server each week at the outset of the program to more than 80 servers each week at the start of the second month. The CIO's full

support as a single-threaded leader was critical to the rapid scale up of servers being migrated. The CIO attended weekly migration cutover calls with the migration team to ensure real-time escalation and resolution of issues, which accelerated the migration velocity.

Align the senior leadership team

It's important to create alignment between the various teams regarding the success criteria of the migration. While migration planning and implementation can be accomplished by a small, dedicated team, challenges arise when defining the strategy and performing peripheral activities. These potential obstacles might require actions or escalations from different areas of the IT organization, including the following:

- Business
- Applications
- Networking
- Security
- Infrastructure
- Third-party vendors

Direct action from application owners, leadership, alignment, and a clear escalation to the single-threaded leader become important.

Team collaboration and ownership

In this section:

Create a cross-functional cloud-enablement team

A critical first step in a large migration project is to enable the organization to work in the cloud. To accomplish this, we recommend building a Cloud Enablement Engine (CEE). The CEE is an empowered and accountable team focused on organization's operational readiness for migrations to AWS. The CEE should be a cross-functional team that includes representation from

infrastructure, applications, operations, and security. The team is charged with the following responsibilities:

- Developing policies
- Defining and implementing tools, processes and the architectures that will establish the organizations cloud operations model
- Continuing to facilitate stakeholder alignment across all the areas that they represent

One healthcare customer didn't start with a CEE. However, through initial pilot migrations, the gap was identified. Leading up to the final migration cutover date, with stringent deadlines in place, the team implemented a *migration war room*. In the migration war room, stakeholders from infrastructure, security, applications, and business could assist in resolving issues.

Define requirements for teams and individuals outside the core migration team in advance

Identify teams and individuals that are outside the core program, and define their involvement during the migration planning phases. To facilitate the momentum of the migration during the later stages, pay specific attention to the application teams' involvement. Their knowledge of the application, ability to diagnose issues, and requirement to sign off on the cutover will be required.

While the migration will be led by a core team, the application teams will likely be involved in validating the migration plan and testing during cutover. Customers often approach the cloud migration as an infrastructure project, instead of as an application migration. This can lead to issues during the migration.

We recommend considering the application team's required involvement when selecting a migration strategy. For example, a rehost strategy requires less application-team involvement compared with a replatform or refactor strategy in which more of the application landscape is being changed. If application owner availability is limited, consider using rehost or replatform as opposed to the refactor, relocate, or repurchase strategies.

Validate that there are no licensing issues when migrating workloads

Licensing might change when you migrate corporate off–the-shelf products to the cloud. Your license agreements might be focused on your on-premises estate. For example, a license might be by CPU or linked to a specific MAC address. Alternatively, license agreements might not include

the right to host in a public cloud environment. However, renegotiating licensing with vendors can include long lead times and presents a hard blocker for the migration.

We recommend collaborating with your sourcing or vendor management teams as soon as the scope of the migration is defined. Licensing might also influence your target architecture and migration patterns.

Training

In this section:

Train teams on new tooling and processes

After the migration strategy is defined, invest time in understanding what training might be required for the migration and for your target operating model. During the migration, you will likely use tooling, such as AWS Database Migration Service, that is new to your organization. Proactively training teams reduces the delays experienced during the migration phases.

We recommend seeking active knowledge transfer methods that provide an opportunity to experiment with the tooling in a hands-on fashion. As an example, AWS Professional Services provided several Cloud Migration Factory training sessions for three systems integrator (SI) AWS Partners responsible for a large migration. This ensured that the team had basic familiarity as it moved into the migration phase. It also helped identify subject matter experts (SMEs) who could serve as first-line escalation within each SI AWS Partner team.

Technology perspective

Technology provides a great foundation for accelerating large migrations. For example, the Cloud Migration Factory solution is focused on how to provide end-to-end automation for migrations. This section explores some of the best practices for using technology to achieve the scale and velocity required, aligned with the scope, strategy, and timelines.

The overarching principle is to look at areas of automation wherever possible. If you have thousands of servers in scope, performing tasks manually can be a costly and time-consuming effort.

To perform a migration, several tools are typically used, such as the following:

- Discovery
- Migration implementation
- Configuration management database (CMDB)
- Inventory spreadsheet
- Project management

These tools are used at different stages of migrations, from assessment to mobilize through to implementation. Selection of these tools is driven by the business objectives and timelines.

After migration phases are planned, the next step is to ensure that the migration team has the skills to use the tools they will need. If a team lacks the skills or experience, plan targeted trainings to ramp up the skill set. If possible, create events where teams can get experience with the migration tooling in a safe environment. For example, are there sandpit or lab servers that teams can migrate to experience with the tooling? Alternatively, is it acceptable for initial development workloads to be used for learning purposes?

Automation, tracking, and tooling integration

In this section:

Automate migration discovery to reduce the time required

Most large migration programs commence by understanding the scope of the migration (what must be migrated) and developing a strategy (how it will be migrated). Discovery is an important aspect of this. The required metadata points are captured to form a migration strategy decision tree. To migrate workloads at pace, you must identify and import the required migration metadata into your implementation processes, such as a migration factory. A fully automated mechanism to extract, transform, load (ETL) the migration metadata greatly reduces the time and level of effort involved in the discovery process.

One customer developed a fully automated data intake process for their migration factory. The migration wave plan with all the migration metadata was hosted and maintained in a spreadsheet

on Microsoft SharePoint. When changes were made to the source, an AWS Lambda function was initiated to load the data into the migration factory without manual intervention. This automated data intake process helped the customer reduce manual work, minimize human error, and accelerate their velocity. They were able to migrate more than 1,000 servers to AWS.

Automate repetitive tasks

In the migration implementation phase, many small processes must be repeated frequently. When using AWS Application Migration Service (MGN), for example, you must install the agent on each server that is in scope of the migration.

Building a migration factory that works for your specific business and technical requirements is the most effective way to achieve the efficiency and velocity required to deliver a successful large migration. A migration factory provides an integration and orchestration framework that uses a standardized dataset to accelerate the migration. After all the tasks are identified, spend time on automating all the manual tasks that can be automated alongside prescriptive runbooks.

The Cloud Migration Factory solution is an example of this. Cloud Migration Factory is designed to provide the migration automation foundations on which you can automate aspects that are specific to your organization. For example, you might want to update a flag in your CMDB to highlight that the on-premises servers can now be decommissioned. In this scenario, you could create an automation that which performs this task at the end of the migration wave. Cloud Migration Factory has a centralized metadata store with all the wave, application, and server metadata. The automation script can connect to Cloud Migration Factory to get a list of servers in that wave and perform any actions accordingly. Cloud Migration Factory supports AWS Application Migration Service.

Automate tracking and reporting to speed decision making

We recommend building an automated migration reporting dashboard to track and report live data, including key performance indicators (KPIs) for the program. Migration projects involve stakeholders from across the organization, including the following:

- Application teams
- Testers
- Decommissioning teams
- Architects
- Infrastructure teams

- Leadership

To perform their roles, these stakeholders require live data. For example, network teams must know the upcoming migration waves to understand the impact on the shared connection between on-premises resources and AWS. Leadership teams want to understand how much of the migration is complete. Having a dependable, automated live feed of data prevents miscommunications and provides a basis on which decisions can be made.

A large healthcare customer was working toward a data center exit with an upcoming deadline. Given the scale and complexity, a significant amount of time was initially spent on tracking and communicating the migration status between stakeholders. The migration team later used Amazon QuickSight to build automated dashboards that visualized the data, significantly simplifying tracking and communications while increasing the migration velocity.

Explore tooling that can facilitate your migration

Choosing the right tools for your migration is not easy, especially if no one in your organization has managed a large migration before.

We recommend spending time to choose suitable tooling to support the migration. This exploration might involve a license cost, but it can provide a cost benefit when you consider the wider initiative. Alternatively, you might find that tooling embedded in your organization can provide a similar outcome. For example, you might already have application performance monitoring tooling deployed across your estate, which can provide rich discovery information.

A technology customer was initially reluctant to run automated discovery tooling during their migration due to a lack of familiarity. As a result, an SI AWS Partner had to run 5#10 hours of meetings per application to discovery the estate manually, including server names, operating system versions, and dependencies. It was estimated that if discovery tooling had been used, the discovery effort could have been reduced by more than 1,000 hours.

Prerequisites and post migration validation

In this section:

Build the landing zone during the pre-migration phase

We recommend building the AWS target environment, or landing zone, ahead of time, instead of building the target virtual private clouds (VPCs) and subnets during the migration wave. Building a well-architected landing zone is a prerequisite for the migration. The landing zone should include monitoring, governance, operational, and security controls.

Building and validating the landing zone ahead of the migration minimizes the uncertainty that comes with running your workloads in a new environment. With the landing zone in place, the stakeholders can focus on migrating the workloads without worrying about aspects managed at an account or VPC level.

Outline prerequisite activities

Alongside the landing zone, it's important to align other technical prerequisites before the migration, especially processes with lengthy lead time. For example, make the necessary firewall changes to allow the data to be replicated from on premises to AWS. Communicating technical prerequisites early helps to prepare and allocate the resources required. It's common for migrations to stall because prerequisites haven't been met. Not only does this impact the in-progress migration wave, it might push back the dates of all future migrations while the issue is being remediated.

A financial services company intended to perform a mass-migration to AWS, with the goal of vacating several data centers. However, their bandwidth available between on-premises and AWS was not sufficient for the velocity they intended. Unfortunately, increasing the bandwidth required a new connection and had a lead time of three months. This meant that the migration velocity was constrained for the first three months.

Implement post-migration checks for continuing improvement

Finally, remember to implement post-migration validations such as operations integration, cost optimization, and governance and compliance checks. Post-migration validation includes assessing previously migrated workloads to uncover technical lessons learned that should be applied to future waves.

Further, this is a great opportunity to implement cost-control operations. For example, during the migration you might decide to size match the AWS instances to your on-premises estate to reduce the need for performance testing. Now that testing is no longer on the data center closure critical path, you can use Amazon CloudWatch to assess the instance utilization and determine whether a smaller-sized instance would be suitable.

To illustrate the importance of this phase, a large technology customer was performing a large migration but initially did not include post-migration validations. After migrating more than 100 servers, they identified that the AWS Systems Manager Agent (SSM Agent) was not configured correctly. All previously migrated servers had to be remediated, and the migration stalled. The customer also identified that the instances were as large as five times the initial estimates, so they implemented a cost checkpoint at the end of each migration wave.

Process perspective

Processes bring consistency but they also evolve and are susceptible to change because each project is unique. As you run the process repeatedly, you will identify gaps and room for improvements that can add up to huge benefits as you fail, learn, adopt, and iterate. These changes can lead to new ideas or innovations that the project and the business can take advantage of in the future, which provides a catalyst for growth that brings quality and team confidence.

Processes in migrations can be complex as they cross technologies and boundaries that might not have been linked previously. This perspective provides processes and guidance on specific requirements for large migrations.

Preparing for your large migration

The following sections outline the core principles that are required to ensure that you start your migration journey with a clear direction and buy-in from the stakeholders that will be critical to its success.

In this section:

Define business drivers and communicate timeline, scope, and strategy

When approaching a large migration to AWS, you will quickly discover that there are numerous ways to migrate your servers. For example, you could do the following:

- Rehost workloads using AWS Application Migration Service.
- Containerize your application and host it on the Amazon Elastic Container Service (Amazon ECS) or the Amazon Elastic Kubernetes Service (Amazon EKS) managed container platform.
- Redesign your workload into a fully serverless application.

To determine the correct migration path, it's important to work backwards from your business drivers. If your ultimate goal is to increase business agility, you might favor the second two patterns, which involve more levels of transformation. If your goal is to vacate a data center by the end of the year, you might choose to rehost workloads because of the velocity that rehosting provides.

A large migration typically involves a wide range of stakeholders, including the following:

- Application owners
- Network teams
- Database administrators
- Executive sponsors

It is key to identify the business drivers of the migration and include that list in a document, such as a project charter that members of the migration program can access. Furthermore, create key performance indicators (KPIs) that closely align with your target business outcomes.

For example, one customer wanted to migrate 2,000 servers within 12 months to achieve their target business outcome of vacating their data center. However, their security teams were not aligned toward this goal. The result was several months of technical debates on whether to miss the data center closure date but modernize applications further or to rehost initially to enable the timely data center closure and then modernize applications on AWS.

Define a clear escalation path to help remove the blockers

Large cloud migration programs typically involve a wide range of stakeholders. After all, you're potentially changing applications that have been hosted on premises for several decades. It's common for each of the stakeholders to have conflicting priorities.

While all the priorities might drive value, the program will likely have a limited amount of budget and a defined target outcome. Managing the various stakeholders and focusing on the target business outcomes can be challenging. This challenge is compounded when you multiply it

by the hundreds or thousands of applications that are in scope of the migration. Further, the stakeholders likely report into different leadership teams, which have other priorities. With this in mind, alongside clearly documenting the target business outcomes, it's important to define a clear escalation matrix to help remove blockers. This can save a significant amount of time and help align the various teams toward a common goal.

One example that demonstrates this is a financial services company whose goal was to vacate their primary data center within 12 months. There wasn't a clear mandate or escalation path, which resulted in the stakeholders crafting their desired migration paths, regardless of time and budget constraints. Following an escalation to the CIO, a clear mandate was set and a mechanism was provided for requesting required decisions.

Minimize unnecessary change

Change is good but more changes mean more risks. When the business case for the large migration is approved, there is most likely a target business outcome driving this initiative, such as vacating a data center by a specific date. While it's common for technologists to want to re-write everything to take full advantage of AWS services, this might not be your business goal.

One customer was focused on a two-year migration of the company's entire web-scale infrastructure to AWS. They created a two-week rule as a mechanism to prevent application teams from spending months rewriting their applications. By using the two-week rule, the customer was able to sustain a long-term migration with a consistent cadence when hundreds of applications had to be moved over a multiple-year period. For more information, see the blog post The Two-Week Rule: Refactor Your Applications for the Cloud in 10 Days.

We recommend minimizing any change that doesn't align with the business outcome. Instead, build mechanisms to manage these additional changes in future projects.

Document an end-to-end process early

Document the complete migration process and assignment of ownership in the early stages of a large migration program. This documentation is important in educating all stakeholders about how the migration will run and their roles and responsibilities. The documentation will also help you to understand where issues might occur and to provide updates and iterations of the process as you progress through the migrations.

During the development of the migration project, ensure that any existing processes are understood and that integration points and dependencies documented clearly. Include places where engagement with external process owners will be required, including change requests,

service requests, vendor support, and network and firewall support. After the process is understood, we recommend documenting ownership in a responsible, accountable, consulted, informed (RACI) matrix to track the different activities. To finalize the process, establish a countdown plan by identifying the timelines involved in each step of the migration. The countdown plan generally works backwards from the workload migration cutover date and time.

This documentation approach worked well for a multinational home appliance corporation that migrated to AWS successfully in less than a year and exited four data centers. They had six different organizational teams and multiple third parties involved, which introduced management overhead resulting in back-and-forth decisions and delays in implementation. The AWS Professional Services team, together with the customer and their third parties, identified key processes for the migration activities and documented them with respective owners. The resulting RACI matrix was shared and agreed upon by all involved teams. Using the RACI matrix and an escalation matrix, the customer alleviated the blockers and issues that were creating delays. They were then able to exit the data centers ahead of schedule.

In another example of using RACI and escalation matrices, an insurance firm was able to exit the data center in less than 4 months. The customer understood and implemented a shared responsibility model, and a detailed RACI matrix was followed to track the progress of each process and activity throughout the migration. As a result, the customer was able to migrate over 350 servers in the first 12 weeks of implementation.

Document standard migration patterns and artifacts

Think of this as creating cookie cutters for the implementation. Reusable references, documentation, runbooks, and patterns are the key to scale. These journal the experience, learning, pitfalls, issues, and solutions that future migration projects can reuse and avoid, significantly accelerating the migration. The patterns and artifacts are also an investment that will help improve the process and guide future projects.

For example, one customer was performing a year-long migration where applications were being migrated by three different SI AWS Partners. In the early stages, each AWS Partner was using their own standards, runbooks, and artifacts. This placed numerous stresses on the customer teams, because the same information could be presented to them in different ways. After these early pains, the customer established central ownership of all documentation and artifacts to be used in the migration, with a process for submitting recommended changes. These assets include the following:

- A standard migration process and checklists

- Network diagram style and format standards
- Application architectural and security standards based on business criticality

In addition, changes to any of these documents and standards were sent out to all teams on a weekly cadence, and each partner was required to confirm receipt and adherence to any changes. This greatly improved communication and consistency for the migration project, and when a separate large migration effort in another business unit started, that team was able to adopt the existing process and documents, greatly accelerating their success.

Establish a single source of truth for migration metadata and status

When it comes to planning a large migration, establishing a source of truth is important to keep the various teams aligned and enable data-driven decisions. When you start this journey, you might find numerous data sources that you can use, such as the configuration management database (CMDB), application performance monitoring tooling, inventory lists, and so on.

Alternatively, you might find that there are few data sources and you must create mechanisms to capture the data needed. For example, you might need to use discovery tooling to uncover technical information, and to survey IT leaders to obtain business information.

It's important to aggregate the various data sources into a single dataset that you can use for the migration. You can then use the single source of truth for tracking the migration during the implementation. For example, you can track which servers have been migrated.

A financial services customer that wanted to migrate all workloads to AWS focused on planning the migration with the dataset that had been provided. This dataset had key gaps, such as business criticality and dependency information, so the program started a discovery exercise.

In another example, a company in the same industry moved into migration wave implementation based on an out-of-date understanding of their server infrastructure inventory. They quickly started to see migration numbers decrease because the data was incorrect. In this case, application owners were not understood, which meant that they could not find testers in time. Additionally, the data were not aligned to the decommissioning that their applications teams had completed, so servers were running without being used for a business purpose.

Running your large migration

After you have established your business outcomes and communicated the strategy to the stakeholders, you can move to planning how you carve up the scope of the large migration into

sustainable migration events or waves. The following examples provide key guidance for making the wave plan.

In this section:

Plan migration waves ahead of time to ensure a steady flow

Planning your migration is one of the most important phases of the program. It goes with the saying "if you fail to plan, you plan to fail." Planning migration waves ahead of time allows the project to flow swiftly as the team becomes more proactive to the migration situation. It helps the project scale more easily, and it improves decision making and forecasting as project demands increase and become complex. Planning ahead also improves the team's ability to adapt to changes.

For example, a large financial services customer was working on a data center exit program. Initially, the customer planned the migration waves in a sequential fashion, completing one wave before beginning to plan the next. This approach resulted in less time to prepare. When the stakeholders were informed that their applications were being migrated to AWS, they still had several steps to perform before starting the migration. This added significant delays to the program. After the customer realized this, they implemented a holistic and future-focused migration planning stream where migration waves were planned several months in advance. This provided enough notice for the application teams to perform their pre-migration activities such as notifying AWS Partners, licensing analysis, and so forth. They could then remove those tasks from the program's critical path.

Keep wave implementation and wave planning as separate processes and teams

When wave planning and wave implementation teams are separate, the two processes can work in parallel. With communication and coordination, this avoids slowing down the migration because not enough servers or applications are ready to achieve the expected velocity. For example, the

migration team might need to migrate 30 servers each week, but only 10 servers are ready in the current wave. This challenge is often caused by the following:

- The migration implementation team was not involved in the wave planning, and the data collected in the wave planning phase are not complete. The migration implementation team must collect more server data before starting the wave.
- Migration implementation is scheduled to start right after wave planning, with no buffer between.

It is critical to plan waves ahead of time, and to create a buffer between preparation and the start of the wave implementation. It is also important to make sure that the wave planning team and the migration team work together to collect the right data and avoid rework.

Start small for great outcomes

Plan to start small and increase migration velocity with each subsequent wave. The initial wave should be a single, small application, fewer than 10 servers. Add additional applications and servers in subsequent waves, building up to your full migration velocity. Prioritizing less complex or risky applications, and ramping velocity on a schedule, gives the team time to adjust to working together and to learn the process. In addition, the team can identify and implement process improvements with each wave, which can substantially improve the velocity of later waves.

One customer was migrating more than 1,300 servers in a year. By starting with a pilot migration and a few smaller waves, the migration team was able to identify multiple ways to improve later migrations. For example, they identified new data center network segments earlier. They worked with their firewall team early in the process to put in place firewall rules that allowed communication with migration tooling. This helped prevent unnecessary delays in future waves. In addition, the team was able to develop scripts to help automate more of their discovery and cutover processes with each wave. Starting small helped the team focus on early process improvements, and greatly increased their confidence.

Minimize the number of cutover windows

Mass migrations require a disciplined approach to driving scale. Being too flexible in some areas is an anti-pattern for large migrations. By limiting the number of weekly cutover windows, time spent on cutover activities has higher value.

For example, if the cutover window is too flexible, you could end up with 20 cutovers with five servers each. Instead, you could have two cutovers with 50 servers each. Because the time and

effort for each cutover are similar, having fewer, larger cutovers reduces the operational burden of scheduling, and limits unnecessary delays.

A large technology company was trying to migrate out of a few leased data centers before contract expiration. Missing the expiration would result in expensive, short-term renewal terms. Earlier in the migration, application teams were allowed to dictate migration schedule up to the last minute, including opting out of migration for any reason just days ahead of cutover. This led to numerous delays in the early stages of the project. Often, the customer had to negotiate with other application teams at the last minute to fill in. The customer eventually increased their planning discipline, but this early mistake led to constant stress for the migration team. Delays to the overall schedule resulted in some applications not making it out of the data centers in time.

Fail fast, apply experience, and iterate

Every migration has pitfalls initially. Failing early helps the team learn, understand the bottlenecks, and apply the lessons learned to larger waves. It is expected that the first couple of waves in a migration will be slow for the following reasons:

- Team members are adjusting to each other and the process.
- Large migrations usually involve many different tools and people.
- It takes time to integrate, test, fail, learn, and continuously improve the end-to-end process.

Issues are common and expected during the first couple of waves. It is important to understand and communicate this to the entire organization, because some teams may not like to try new things and fail. Failure can discourage the team and become a blocker for future migrations. Making sure everyone understands that initial issues are part of the job and encouraging everyone to try and fail is key to a successful migration.

One company planned to migrate more than 10,000 servers in 24–36 months. To achieve that goal, they needed to migrate nearly 300 servers a month. However, that does not mean they migrated 300 servers from day one. The first couple waves were learning waves so that the team could understand how things worked and who had permissions to do what. They also identified integrations that would improve the process, such as integrating with CMDB and CyberArk. They used the learning waves to fail, improve, and fail again, refining the process and automation. After 6 months, they were able to migrate more than 120 servers each week.

Don’t forget the retrospective

This is an important part of an agile process. It’s where the team communicates, adjusts, learns, agrees, and moves forward. A retrospective at the most basic level is looking back, discussing what happened, determining what went well and what needs to improve. Improvements can then be built based on those discussions. Retrospectives wrap some formality or process around the idea of lessons-learned. Retrospectives are important because to achieve the scale and velocity for large migrations to succeed, the processes, tools, and teams must constantly evolve and improve. Retrospectives can play a significant part in that.

Traditional lessons-learned sessions do not happen until the end of a program, so often these lessons do not get reviewed at the start of the next migration wave. With large migrations, lessons learned should be applied to the next wave and should be a key part of the wave planning process.

For one customer, weekly retrospectives were held to discuss and document lessons learned from the cutovers. In these sessions, they uncovered areas where there was scope for streamlining from a process standpoint or automation. This resulted in implementation of a countdown schedule with specific activities, owners, and automation scripts to minimize manual tasks, including validation of third-party tools and Amazon CloudWatch agent installation, during cutover.

At another large tech company, regular retrospectives were held with the team to identify problems with previous migration waves. This resulted in process, script, and automation improvements that drove the average migration time down by 40 percent over the course of the program.

Additional considerations

Many areas must be factored into a large migration program. The following sections provide thoughts on other items that must be considered.

In this section:

Clean up as you go

A migration isn’t considered successful if it costs 10 times what you expected, and the project is not complete until the resources used for migration are shut down and cleaned up. This cleanup

should be part of the post-migration activity. It ensures that you will not leave unused resources and services in your environment that will add to the costs. Post-migration cleanup is also a good security practice for preventing threats and vulnerabilities that expose your environment.

Two key outcomes of moving to the AWS Cloud are the cost savings and the security. Leaving unused resources can defeat the business purpose of moving to cloud. The most common resources that are not cleaned up include the following:

- Test data
- Test databases
- Test accounts, including firewall rules, security groups, and network access control list (network ACL) IP addresses
- Ports provisioned for testing
- Amazon Elastic Block Store (Amazon EBS) volumes
- Snapshots
- Replication (such as stopping the data replication from on-premises to AWS)
- Files that consume space (such as temporary database backups used to migrate)
- Instances that host the migration tools

In one example of bad cleanup practices, SI AWS Partners were not removing replication agents after a successful migration. An AWS audit discovered that replication servers and EBS volumes that had already been migrated were costing $20,000 (USD) each month. To mitigate the issue, AWS Professional Services created an automated audit process that notified SI AWS Partners when stale servers were still being replicated. The SI AWS Partners could then take action on unused and stale instances.

For future migrations, a process was adopted to define a post-migration hypercare period of 48 hours to ensure smooth platform adoption. The customer's infrastructure team then submitted a decommission request for on-premises servers. It was advised that upon approval of the decommission request, servers of the respective wave would be removed from the application migration service console.

Implement multiple phases for any additional transformation

When carrying out a large migration, it's important to remain focused on your core goal, such as data center closure or infrastructure transformation. In smaller migrations, scope creep might have

a minimal impact. However, a few days of additional effort multiplied by potentially thousands of servers can add a significant amount of time to the program. Furthermore, the additional changes might also require updates to documentation, process, and training for support teams.

To overcome potential scope creep, you can implement a multiple-phase approach to your migration. For example, if your goal was to vacate a data center, phase 1 may include only rehosting the workload to AWS as fast as possible. After a workload is rehosted, phase 2 can implement transformational activities without risking the target business outcome.

For example, one customer planned to exit their data center in 12 months. However, their migration encompassed other transformation activities, such as rolling out new application performance monitoring tooling and upgrading operating systems. More than 1,000 servers were in scope of the migration, so these activities added a significant delay to the migration. Furthermore, this approach required training in the use of the new tooling. The customer later decided to implement a multiple-phase approach with an initial focus on rehost. This increased their migration velocity and reduced the risk of not meeting the data center closure date.

Conclusion

Large migrations present different challenges when compared to smaller migrations. This is mostly due to the complexities introduced by the scale. For example, installing an agent onto a single server is fairly straightforward and will take approximately 5 minutes. However, if you have 5,000 servers in scope for your migration, this will take approximately 416 hours and will present the following challenges:

- It's likely that there are multiple operating systems that require different processes.
- There might be separate Microsoft Active Directory domains to manage due to previous mergers and acquisitions.
- Effective processes and tools are required to orchestrate the agent installation for each wave and then track and report the progress.

This strategy outlines large migration best practices based on AWS Professional Services experiences helping a wide range of customers. This includes people, process, and technology perspectives. If you want to start or are in the process of migrating to AWS, consultants at AWS Professional Services would be happy to assist you. Contact your AWS representative to start the conversation.

For next steps, we recommend that you review the AWS Prescriptive Guidance series designed to help you plan and complete a large migration to the AWS Cloud. For the complete series, see Large migrations to the AWS Cloud.

Resources

AWS large migrations

To access the complete AWS Prescriptive Guidance series for large migrations, see Large migrations to the AWS Cloud.

Related AWS Prescriptive Guidance resources

- Automating large-scale server migrations with Cloud Migration Factory
- Best practices for assessing applications to be retired during a migration to the AWS Cloud
- Setting up a secure and scalable multi-account AWS environment
- Evaluating migration readiness
- Mobilize your organization to accelerate large-scale migrations

Additional references

- AWS Cloud Migration Factory solution
- Free cloud migration services on AWS
- AWS Database Migration Service
- Migrate with AWS

Videos

- Executing a large-scale migration to AWS (AWS re:Invent 2020)
- CloudEndure Migration Factory best practices (AWS re:Invent 2020)

Contributors

This strategy was authored by the global Large Migration tiger team within AWS Professional Services. The team has successfully migrated thousands of servers to AWS on behalf of AWS customers. Contributors to this document include:

- Chris Baker, Principal Product Engineer
- Dwayne Bordelon, Senior Cloud Application Architect
- Rodolfo Jr. Cerrada, Senior Application Architect
- Pratik Chunawala, Principal Cloud Architect
- Bill David, Principal Customer Solutions Manager
- Dev Kar, Senior Consultant
- Wally Lu, Principal Consultant
- Jon Madison, Principal Cloud Architect
- Abhishek Naik, Senior Solution Architect
- Damien Renner, Senior Migration Specialist
- Amit Rudraraju, Senior Cloud Architect

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
Removed CloudEndure Migration service	We removed references to the CloudEndure Migration service. AWS Application Migration Service is the primary migration service recommended for lift-and-shift migrations to the AWS Cloud.	May 11, 2022
Updated name of AWS solution	We updated the name of the referenced AWS solution from *CloudEndure Migration Factory* to *Cloud Migration Factory*.	May 2, 2022
Updated resources	We updated the Introduction and Resources sections with the latest documents in the large migration series.	March 8, 2022
Initial publication	—	September 16, 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 AWS CodeCommit. 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, AWS Panorama offers devices that add CV to on-premises camera networks, and Amazon SageMaker 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.

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.

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.

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.

G

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.

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.

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 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.

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

An approach in system engineering that takes privacy into account throughout the whole engineering 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.

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).

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.

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 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.

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.

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.