AWS Key Management Service best practices

AWS Prescriptive Guidance

AWS Prescriptive Guidance: AWS Key Management Service best practices

Table of Contents

AWS Key Management Service best practices

Amazon Web Services (contributors)

March 2025 (document history)

AWS Key Management Service (AWS KMS) is a managed service that makes it easy for you to create and control the cryptographic keys that are used to protect your data. This guide describes how to effectively use AWS KMS and provides best practices. It helps you compare configuration options and choose the best set for your needs.

This guide includes recommendations for how your organization can use AWS KMS to protect sensitive information and implement signing for multiple use cases. It considers current recommendations that use the following dimensions:

- **Managing keys –** Delegation options for management and key storage choices
- **Data protection –** Encrypting data within your own applications vs in AWS services doing it on your behalf
- **Access management –** Using AWS KMS key policies and AWS Identity and Access Management (IAM) policies to implement role-based access control (RBAC) or attribute-based access control (ABAC).
- **Multi-account and multi-Region architecture –** Recommendations for large scale deployments.
- **Billing and cost management –** Understanding your cost and usage, and recommendations for ways to reduce costs.
- **Detective controls –** Monitoring the status of your KMS keys, encryption settings, and encrypted data.
- **Incident response –** Correcting misconfigurations that result in non-compliance with your data protection policies.

Targeted business outcomes

Your data is a critical and sensitive asset for your business. With AWS KMS, you manage the cryptographic keys that are used to protect and verify your data. You control how your data is used, who has access to it, and how it is encrypted. This guide is intended to help developers, system administrators, and security professionals implement encryption best practices that help you secure sensitive data that is stored or transmitted through AWS services. By understanding and implementing the recommendations in this guide, you can promote data confidentiality and integrity across your AWS environment. You can meet your data protection requirements, whether those requirements are formulated internally, or you have requirements specific to a compliance or validation program. For more information about how AWS KMS can help you secure data in your AWS environment, see Using AWS KMS encryption with AWS services in the AWS KMS documentation.

About AWS KMS keys

AWS Key Management Service (AWS KMS) allows you to create cryptographic keys that can be used on data that you pass to the service. The primary resource type is the KMS key, of which there are three types:

- **Advanced Encryption Standard (AES) symmetric keys –** These are 256-bit keys that are used under the Galois Counter Mode (GCM) mode of AES. These keys provide authenticated encryption and decryption of data that is less than 4 KB in size. This is the most common type of key. It is used to protect other data keys, such as those used in your applications or by AWS services that encrypt data on your behalf.
- **RSA or elliptic curve asymmetric keys** -– These keys are available in various sizes and support many algorithms. Depending on the algorithm, they can be used for encryption and decryption and for sign and verify operations.
- **Symmetric keys for performing hash-based message authentication code (HMAC) operations** – These keys are 256-bit keys that are used for sign and verify operations.

KMS keys cannot be exported from the service in plaintext. They are generated by and can only be used within the hardware security modules (HSMs) used by the service. This is a foundational security property of AWS KMS to prevent key compromise. In China (Beijing) and China (Ningxia) Regions, these HSMs are certified by OSCCA. In all other Regions, the HSMs used in AWS KMS are validated under the FIPS 140 program within NIST at Security Level 3. For more information about design and controls in AWS KMS that help protect your keys, see AWS Key Management Service Cryptographic Details.

You can submit data to AWS KMS by using various cryptographic APIs in order to perform encrypt, decrypt, sign, or verify operations with KMS keys. You can also choose to have a KMS key act like a *key-encryption key*, which protects a key type called a *data key*. A data key can be exported from AWS KMS for use within your local application or an AWS service that is protecting data on your behalf. The use of data keys is common in all key management systems and is often referred to as envelope encryption. *Envelope encryption* allows a data key to be used on the remote system that is handling your sensitive data, instead of having to send your sensitive data to AWS KMS for encryption directly under a KMS key.

For more information, see AWS KMS keys and AWS KMS cryptography essentials in the AWS KMS documentation.

Key management best practices for AWS KMS

When using AWS Key Management Service (AWS KMS), there are some fundamental design decisions that you must make. These include whether to use a centralized or decentralized model for key management and access, the type of keys to use, and the type of key store to use. The following sections help you make decisions that are right for your organization and use cases. This section concludes with important considerations for disabling and deleting KMS keys, including actions that you should take to help protect your data and keys.

This section contains the following topics:

Choosing a centralized or decentralized model

AWS recommends that you use multiple AWS accounts and manage those accounts as a single organization in AWS Organizations. There are two broad approaches to managing AWS KMS keys in multi-account environments.

The first approach is a decentralized approach, where you create keys in each account that uses those keys. When you store the KMS keys in the same accounts as the resources they protect, it is easier to delegate permissions to local administrators who understand access requirements for their AWS principals and keys. You can authorize key usage by using just a key policy, or you can combine a key policy and identity-based policies in AWS Identity and Access Management (IAM).

The second approach is a centralized approach, where you maintain KMS keys in one or a few designated AWS accounts. You allow other accounts to only use the keys for cryptographic operations. You manage keys, their life cycle, and their permissions from the centralized account. You permit other AWS accounts to use the key but don't allow other permissions. The external accounts cannot manage anything about the key's life cycle or access permission. This centralized model can help minimize the risk of unintended deletion of keys or privilege escalation by delegated administrators or users.

The option you choose depends on several factors. Consider the following when choosing an approach:

1. Do you have an automated or manual process for provisioning key and resource access? This includes resources such as deployment pipelines and infrastructure as code (IaC) templates. These tools can help you deploy and manage resources (such as KMS keys, key policies, IAM roles, and IAM policies) across many AWS accounts. If you don't have these deployment tools, a centralized approach to key management might be more manageable for your business.
2. Do you have administrative control over all of the AWS accounts that contain resources that use KMS keys? If so, a centralized model can simplify management and eliminate the need to switch AWS accounts to manage keys. Note, however, that IAM roles and user permissions to use keys still must be managed per account.
3. Do you need to offer access to use your KMS keys to customers or partners who have their own AWS accounts and resources? For these keys, a centralized approach can reduce administrative burden on your customers and partners.
4. Do you have authorization requirements for access to AWS resources that are better solved by either a centralized or local access approach? For example, if different applications or business units are responsible for managing the security of their own data, a decentralized approach to key management is better.
5. Are you exceeding service resource quotas for AWS KMS? Because these quotas are set per AWS account, a decentralized model distributes the load across accounts, effectively multiplying service quotas.

> **Note**
>
> The management model for keys is irrelevant when considering request quotas because these quotas are applied to the account principal making a request against the key, not the account that owns or manages the key.

In general, we recommend that you start with a decentralized approach unless you can articulate a need for a centralized KMS key model.

Choosing customer managed keys, AWS managed keys, or AWS owned keys

The KMS keys that you create and manage for use in your own cryptographic applications are known as *customer managed keys*. AWS services can use customer managed keys to encrypt the data the service stores on your behalf. Customer managed keys are recommended if you want full control over the life cycle and usage of your keys. There is a monthly cost to have a customer managed key in your account. In addition, requests to use or manage the key incur a usage cost. For more information, see AWS KMS pricing.

If you want an AWS service to encrypt your data but don't want the overhead or costs of managing keys, you can use an *AWS managed key*. This type of key exists in your account, but it can only be used under certain circumstances. It can only be used in the context of the AWS service that you're operating in, and it can only be used by principals within the account that contains the key. You cannot manage anything about the life cycle or permissions of these keys. Some AWS services use AWS managed keys. The format of an AWS managed key alias is `aws/<service code>`. For example, an `aws/ebs` key can only be used to encrypt Amazon Elastic Block Store (Amazon EBS) volumes in the same account as the key and can only be used by IAM principals in that account. An AWS managed key can be used only by users in that account and for resources in that account. You cannot share resources encrypted under an AWS managed key with other accounts. If this is a limitation for your use case, we recommend using a customer managed key instead; you can share use of that key with any other account. You are not charged for the existence of an AWS managed key in your account, but you are charged for any use of this key type by the AWS service that is assigned to the key.

An AWS managed key is a legacy key type that is no longer being created for new AWS services as of 2021. Instead, new (and legacy) AWS services are using an *AWS owned key* to encrypt your data by default. AWS owned keys are a collection of KMS keys that an AWS service owns and manages for use in multiple AWS accounts. Although these keys are not in your AWS account, an AWS service can use one to protect the resources in your account.

We recommend that you use customer managed keys when granular control is most important and use AWS owned keys when convenience is most important.

The following table describes the key policy, logging, management, and pricing differences between each key type. For more information about key types, see AWS KMS concepts.

Consideration	Customer managed keys	AWS managed keys	AWS owned keys
Key policy	Exclusively controlled by the customer	Controlled by service; viewable by customer	Exclusively controlled and only viewable by the AWS service that encrypts your data
Logging	AWS CloudTrail customer trail or event data store	CloudTrail customer trail or event data store	Not viewable by the customer
Life cycle management	Customer manages rotation, deletion, and AWS Region	AWS service manages rotation (annual), deletion, and Region	AWS service manages rotation (annual), deletion, and Region
Pricing	Monthly fee for key existence (pro-rated hourly); the caller is charged for API usage	No charge for key existence; the caller is charged for API usage	No charges to customer

Choosing an AWS KMS key store

A *key store* is a secure location for storing and using cryptographic key material. The industry best practice for key stores is to use a device known as a hardware security module (HSM) that has been validated under the NIST Federal Information Processing Standards (FIPS) 140 Cryptographic Module Validation Program at Security Level 3. There are other programs to support key stores that are used to process payments. AWS Payment Cryptography is a service you can use to protect data related to your payment workloads.

AWS KMS supports multiple key store types to help protect your key material when using AWS KMS to create and manage your encryption keys. All of the key store options supplied by AWS KMS are continually validated under FIPS 140 at Security Level 3. They are designed to prevent anyone, including AWS operators, from accessing your plaintext keys or using them without your permission. For more information about the available types of key stores, see Key stores in the AWS KMS documentation.

The AWS KMS standard key store is the best choice for the majority of workloads. If you need to choose a different type of key store, carefully consider whether regulatory or other requirements (such as internal) mandate making this choice, and carefully weigh the costs and benefits.

Deleting and disabling KMS keys

The deletion of a KMS key can have significant impact. Before you delete a KMS key that you no longer intend to use, consider whether it's adequate to set the key state to **Disabled**. While a key is disabled, it cannot be used for cryptographic operations. It still exists in AWS, and you can re-enable it in the future if required. Disabled keys continue to incur storage charges. We recommend that you disable keys instead of deleting them until you are confident that the key does not protect any data or data keys.

> ⚠ **Important**
>
> Deleting a key must be carefully planned. Data cannot be decrypted if the corresponding key has been deleted. AWS has no means to recover a deleted key after it's been deleted. As with other critical operations in AWS, you should apply a policy that limits who can schedule keys for deletion and require multi-factor authentication (MFA) for key deletion.

To help prevent accidental key deletion, AWS KMS enforces a default minimum waiting period of seven days after the execution of a `DeleteKey` call before it deletes the key. You can set the waiting period to a maximum value of 30 days. During the waiting period, the key is still stored in AWS KMS in a **Pending Deletion** state. It can't be used for encrypt or decrypt operations. Any attempt to use a key that is in the **Pending Deletion** state for encryption or decryption is logged to AWS CloudTrail. You can set an Amazon CloudWatch alarm for these events in your CloudTrail logs. If you receive alarms on these events, you can choose to cancel the deletion process if needed. Until the waiting period has expired, you can recover the key from the **Pending Deletion** state and restore it to either a **Disabled** or **Enabled** state.

Deletion of a multi-Region key requires that you delete replicas before the original copy. For more information, see Deleting multi-Region keys.

If you are using a key with imported key material, you can delete the imported key material immediately. This is different from deleting a KMS key in several ways. When you perform the `DeleteImportedKeyMaterial` action, AWS KMS deletes the key material, and the key state changes to **Pending import**. After you delete the key material, the key is immediately unusable.

There is no waiting period. To enable use of the key again, you need to import the same key material again. The waiting period for KMS key deletion also applies to KMS keys with imported key material.

If data keys are protected by a KMS key and are actively in use by AWS services, they are not immediately affected if their associated KMS key is disabled or if its imported key material is deleted. For example, say that a key with imported material was used to encrypt an object with SSE-KMS. You are uploading the object to an Amazon Simple Storage Service (Amazon S3) bucket. Before you upload the object to the bucket, you import the material into your key. After the object is uploaded, you delete the imported key material from that key. The object remains in the bucket in an encrypted state, but no one can access the object until the deleted key material is re-imported into the key. Though this flow requires precise automation for importing and deleting key material from a key, it can provide an additional level of control within an environment.

AWS offers prescriptive guidance to help you monitor and remediate (if necessary) the scheduled deletion of KMS keys. For more information, see Monitor and remediate scheduled deletion of AWS KMS keys.

Data protection best practices for AWS KMS

This section helps you to make choices about AWS Key Management Service (AWS KMS) key usage for data protection, such as which keys to use for each data type. It also provides specific examples of using AWS KMS with different AWS services. These recommendations and examples help you understand how many keys you might need and which principals require permissions to use those keys.

The section also discusses key rotation. *Key rotation* is the practice of either replacing an existing KMS key with a new key or replacing the cryptographic material associated with an existing KMS key with new material. This guide provides examples and instructions for how to rotate KMS keys for commonly used AWS services. The recommendations and examples are designed to help you make informed choices about your key rotation strategy.

Finally, this section makes recommendations for how to use the AWS Encryption SDK, a tool for implementing client-side encryption in your applications. This section includes design choices that you can make based on the feature set and capabilities of the AWS Encryption SDK.

This section discusses the following encryption topics:

Encryption with AWS KMS

Encryption is a general best practice to protect the confidentiality and integrity of sensitive information. You should use your existing data classification levels and have at least one AWS Key Management Service (AWS KMS) key per level. For example, you could define a KMS key for data classified as **Confidential**, one for **Internal-Only**, and one for **Sensitive**. This helps you make sure that only authorized users have permissions to use the keys associated with each classification level.

Note

A single customer managed KMS key can be used across any combination of AWS services or your own applications that store data of a particular classification. The limiting factor in using a key across multiple workloads and AWS services is how complex the usage permissions need to be to control access to the data across a set of users. The AWS KMS key policy JSON document must be less than 32 KB. If this size restriction becomes a limitation, consider using AWS KMS grants or creating multiple keys to minimize the size of the key policy document.

Instead of relying only on data classification to partition your KMS key, you could also choose to assign a KMS key to be used for a data classification within a single AWS service. For example, all data tagged `Sensitive` in Amazon Simple Storage Service (Amazon S3) should be encrypted under a KMS key that has a name like `S3-Sensitive`. You can further distribute your data across multiple KMS keys within your defined data classification and AWS service or application. For example, you might be able to delete some datasets in a specific time-period and delete other datasets in a different time period. You can use resource tags to help you identify and sort data that is encrypted with specific KMS keys.

If you choose a decentralized management model for KMS keys, you should apply guardrails to make sure that new resources with a given classification are created and use the expected KMS keys with the right permissions. For more information about how you can enforce, detect, and manage resource configuration by using automation, see the Detection and monitoring section of this guide.

This section discusses the following encryption topics:

- Log data encryption with AWS KMS
- Encryption by default
- Database encryption with AWS KMS
- PCI DSS data encryption with AWS KMS
- Using KMS keys with Amazon EC2 Auto Scaling

Log data encryption with AWS KMS

Many AWS services, such as Amazon GuardDuty and AWS CloudTrail, offer options to encrypt log data that is sent to Amazon S3. When exporting findings from GuardDuty to Amazon S3, you must use a KMS key. We recommend that you encrypt all log data and granting decrypt access only to authorized principals, such as security teams, incident responders, and auditors.

The AWS Security Reference Architecture recommends creating a central AWS account for logging. When you do this, you can also reduce your key management overhead. For example, with

CloudTrail, you can create an organization trail or event data store to log events across your organization. When you configure your organizational trail or event data store, you can specify a single Amazon S3 bucket and KMS key in your designated logging account. This configuration applies to all member accounts in the organization. All accounts then send their CloudTrail logs to the Amazon S3 bucket in the logging account, and the log data is encrypted with the specified KMS key. You need to update the key policy for this KMS key to grant CloudTrail the necessary permissions to use the key. For more information, see Configure AWS KMS key policies for CloudTrail in the CloudTrail documentation.

To help protect the GuardDuty and CloudTrail logs, the Amazon S3 bucket and KMS key must be in the same AWS Region. The AWS Security Reference Architecture also provides guidance on logging and multi-account architectures. When aggregating logs across multiple Regions and accounts, review Creating a trail for an organization in the CloudTrail documentation to learn more about opt-in Regions and make sure that your centralized logging works as designed.

Encryption by default

AWS services that store or process data typically offer encryption at rest. This security feature helps protect your data by encrypting it when it's not in use. Authorized users can still access it when needed.

The implementation and encryption options vary between AWS services. Many provide encryption by default. It's important to understand how encryption works for each service that you use. The following are some examples:

- **Amazon Elastic Block Store (Amazon EBS) –** When you enable encryption by default, all new Amazon EBS volumes and snapshot copies are encrypted. AWS Identity and Access Management (IAM) roles or users can't launch instances with unencrypted volumes or volumes that don't support encryption. This feature helps with security, compliance, and auditing by making sure that all data stored on Amazon EBS volumes is encrypted. For more information about encryption in this service, see Amazon EBS encryption in the Amazon EBS documentation.
- **Amazon Simple Storage Service (Amazon S3) –** All new objects are encrypted by default. Amazon S3 automatically applies server-side encryption with Amazon S3 managed keys (SSE-S3) for each new object, unless you specify a different encryption option. IAM principals can still upload unencrypted objects to Amazon S3 by explicitly stating so in the API call. In Amazon S3, to enforce SSE-KMS encryption, you must use a bucket policy with conditions that require encryption. For a sample policy, see Require SSE-KMS for all objects written to a bucket in the Amazon S3 documentation. Some Amazon S3 buckets receive and serve large numbers of

objects. If those objects are encrypted with KMS keys, a large number of Amazon S3 operations results in a large number of GenerateDataKey and Decrypt calls to AWS KMS. This can increase the charges you incur for AWS KMS usage. You can configure Amazon S3 bucket keys, which may significantly reduce your AWS KMS costs. For more information about encryption in this service, see Protecting data with encryption in the Amazon S3 documentation.

- **Amazon DynamoDB** – DynamoDB is a fully managed NoSQL database service that enables server-side encryption at rest by default, and you can't disable it. We recommend that you use a customer managed key for encrypting your DynamoDB tables. This approach helps you implement least privilege with granular permissions and separation of duties by targeting specific IAM users and roles in your AWS KMS key policies. You can also choose AWS managed or AWS owned keys when configuring encryption settings for your DynamoDB tables. For data requiring a high degree of protection (where data should only be visible as cleartext to the client), consider using client-side encryption with the AWS Database Encryption SDK. For more information about encryption in this service, see Data protection in the DynamoDB documentation.

Database encryption with AWS KMS

The level at which you implement encryption affects database functionality. The following are the tradeoffs that you must consider:

- If you use only AWS KMS encryption, the storage that backs your tables is encrypted for DynamoDB and Amazon Relational Database Service (Amazon RDS). This means that the operating system that runs the database sees the contents of the storage as cleartext. All database functions, including index generation and other higher-order functions that require access to the cleartext data, continue to work as expected.
- Amazon RDS builds on Amazon Elastic Block Store (Amazon EBS) encryption to provide full disk encryption for database volumes. When you create an encrypted database instance with Amazon RDS, Amazon RDS creates an encrypted Amazon EBS volume on your behalf to store the database. Data stored at rest on the volume, database snapshots, automated backups, and read replicas are all encrypted under the KMS key that you specified when you created the database instance.
- Amazon Redshift integrates with AWS KMS and creates a four-tier hierarchy of keys that are used to encrypt the cluster level through the data level. When you launch your cluster, you can choose to use AWS KMS encryption. Only the Amazon Redshift application and users with appropriate permissions can see cleartext when the tables are opened (and decrypted) in memory. This is

broadly analogous to the *transparent* or *table-based data encryption (TDE)* features that are available in some commercial databases. This means that all database functions, including index generation and other higher-order functions that require access to the cleartext data, continue to work as expected.

- The client-side data-level encryption implemented through the AWS Database Encryption SDK (and similar tools) means that both the operating system and the database see only ciphertext. Users can view cleartext only if they access the database from a client that has the AWS Database Encryption SDK installed and they have access to the relevant key. Higher-order database functions that require access to cleartext to work as intended—such as index generation—won't work if directed to operate on encrypted fields. When choosing to use client-side encryption, make sure that you use a robust encryption mechanism that helps prevent common attacks against encrypted data. This includes using a strong encryption algorithm and appropriate techniques, such as a salt, to help mitigate ciphertext attacks.

We recommend using the AWS KMS integrated encryption capabilities for AWS database services. For workloads that process sensitive data, client-side encryption should be considered for the sensitive data fields. When using client-side encryption, you should consider the impact to database access, such as joins within SQL queries or index creation.

PCI DSS data encryption with AWS KMS

The security and quality controls in AWS KMS have been validated and certified to meet the requirements of the Payment Card Industry Data Security Standard (PCI DSS). This means that you can encrypt primary account number (PAN) data with a KMS key. The use of a KMS key to encrypt data removes some of the burden of managing encryption libraries. Additionally, KMS keys cannot be exported from AWS KMS, which reduces concern about the encryption keys being stored in an unsecure manner.

There are other ways you can use AWS KMS to meet PCI DSS requirements. For example, if you are using AWS KMS with Amazon S3, you can store PAN data in Amazon S3 because the access control mechanism for each service is distinct from the other.

As always, when reviewing your compliance requirements, make sure that you obtain advice from suitably experienced, qualified, and verified parties. Be aware of the AWS KMS request quotas when you design applications that use the key directly to protect card transaction data that is in scope of the PCI DSS.

Because all AWS KMS requests are logged in AWS CloudTrail, you can audit key usage by reviewing the CloudTrail logs. However, if you use Amazon S3 bucket keys, there is no entry that corresponds to every Amazon S3 action. This is because the bucket key encrypts the data keys that you use to encrypt the objects in Amazon S3. While the use of a bucket key does not eliminate all API calls to AWS KMS, it reduces the number of them. As a result, there is no longer a one-to-one match between Amazon S3 object access attempts and API calls to AWS KMS.

Using KMS keys with Amazon EC2 Auto Scaling

Amazon EC2 Auto Scaling is a recommended service for automating the scaling of your Amazon EC2 instances. It helps you make sure that you have the correct number of instances available to handle the load for your application. Amazon EC2 Auto Scaling uses a service-linked role that provides appropriate permissions to the service and authorizes its activities within your account. To use KMS keys with Amazon EC2 Auto Scaling, your AWS KMS key policies must allow the service linked role to use your KMS key with some API operations, such as `Decrypt`, for the automation to be useful. If the AWS KMS key policy does not authorize the IAM principal who is performing the operation to conduct an action, that action will be denied. For more information about how to correctly apply permissions in the key policy to allow access, see Data protection in Amazon EC2 Auto Scaling in the Amazon EC2 Auto Scaling documentation.

Key rotation for AWS KMS and scope of impact

We do not recommend AWS Key Management Service (AWS KMS) key rotation unless you are required to rotate keys for regulatory compliance. For example, you might be required to rotate your KMS keys due to business policies, contract rules, or government regulations. The design of AWS KMS significantly reduces the types of risk that key rotation is typically used to mitigate. If you must rotate KMS keys, we recommend that you use automatic key rotation and use manual key rotation only if automatic key rotation is not supported.

This section discusses the following key rotation topics:

- AWS KMS symmetric key rotation
- Key rotation for Amazon EBS volumes
- Key rotation for Amazon RDS
- Key rotation for Amazon S3 and Same-Region Replication
- Rotating KMS keys with imported material

AWS KMS symmetric key rotation

AWS KMS supports automatic key rotation only for symmetric encryption KMS keys with key material that AWS KMS creates. Automatic rotation is optional for customer managed KMS keys. On an annual basis, AWS KMS rotates the key material for AWS managed KMS keys. AWS KMS saves all previous versions of the cryptographic material in perpetuity, so you can decrypt any data that is encrypted with that KMS key. AWS KMS does not delete any rotated key material until you delete the KMS key. Also, when you decrypt an object by using AWS KMS, the service determines the correct backing material to use for the decrypt operation; no additional input parameters need to be supplied.

Because AWS KMS retains previous versions of the cryptographic key material and because you can use that material to decrypt data, key rotation doesn't provide any additional security benefits. The key rotation mechanism exists to make it easier to rotate keys if you are operating a workload in a context where regulatory or other requirements demand it.

Key rotation for Amazon EBS volumes

You can rotate Amazon Elastic Block Store (Amazon EBS) data keys by using one of the following approaches. The approach depends on your workflows, deployment methods, and application architecture. You might want to do this when changing from an AWS managed key to a customer managed key.

To use operating system tools to copy the data from one volume to another

1. Create the new KMS key. For instructions, see Create a KMS key.
2. Create a new Amazon EBS volume that is the same size as or larger than the original. For encryption, specify the KMS key that you created. For instructions, see Create an Amazon EBS volume.
3. Mount the new volume on the same instance or container as the original volume. For instructions, see Attach an Amazon EBS volume to an Amazon EC2 instance.
4. Using your preferred operating system tool, copy data from the existing volume to the new volume.
5. When the sync is complete, during a pre-scheduled maintenance window, stop the traffic to the instance. For instructions, see Manually stop and start your instances.
6. Unmount the original volume. For instructions, see Detach an Amazon EBS volume from an Amazon EC2 instance.

7. Mount the new volume to the original mount point.
8. Verify that the new volume is operating correctly.
9. Delete the original volume. For instructions, see Delete an Amazon EBS volume.

To use an Amazon EBS snapshot to copy the data from one volume to another

1. Create the new KMS key. For instructions, see Create a KMS key.
2. Create an Amazon EBS snapshot of the original volume. For instructions, see Create Amazon EBS snapshots.
3. Create a new volume from the snapshot. For encryption, specify the new KMS key that you created. For instructions, see Create an Amazon EBS volume.

   > **Note**
   >
   > Depending on your workload, you might want to use Amazon EBS fast snapshot restore to minimize initial latency on the volume.

4. Create a new Amazon EC2 instance. For instructions, see Launch an Amazon EC2 instance.
5. Attach the volume that you created to the Amazon EC2 instance. For instructions, see Attach an Amazon EBS volume to an Amazon EC2 instance.
6. Rotate the new instance into production.
7. Rotate the original instance out of production and delete it. For instructions, see Delete an Amazon EBS volume.

> **Note**
>
> It is possible to copy snapshots and modify the encryption key used for the target copy. After you copy the snapshot and encrypt it with your preferred KMS keys, you can also create an Amazon Machine Image (AMI) from snapshots. For more information, see Amazon EBS encryption in the Amazon EC2 documentation.

Key rotation for Amazon RDS

For some services, such as Amazon Relational Database Service (Amazon RDS), data encryption occurs within the service and is provided by AWS KMS. Use the following instructions to rotate a key for an Amazon RDS database instance.

To rotate a KMS key for an Amazon RDS database

1. Create a snapshot of the original encrypted database. For instructions, see Managing manual backups in the Amazon RDS documentation.
2. Copy the snapshot to a new snapshot. For encryption, specify the new KMS key. For instructions, see Copying a DB snapshot for Amazon RDS.
3. Use the new snapshot to create a new Amazon RDS cluster. For instructions, see Restoring to a DB instance in the Amazon RDS documentation. By default, the cluster uses the new KMS key.
4. Verify the operation of the new database and the data in it.
5. Rotate the new database into production.
6. Rotate the old database out of production and delete it. For instructions, see Deleting a DB instance.

Key rotation for Amazon S3 and Same-Region Replication

For Amazon Simple Storage Service (Amazon S3), to change the encryption key of an object, you need to read and rewrite the object. When you rewrite the object, you explicitly specify the new encryption key in the write operation. To do this for many objects, you can use Amazon S3 Batch Operations. Within the job settings, for the copy operation, specify the new encryption settings. For example, you might choose **SSE-KMS** and enter the **keyId**.

Alternatively, you could use Amazon S3 Same-Region Replication (SRR). SSR can re-encrypt the objects in transit.

Rotating KMS keys with imported material

AWS KMS does not recover or rotate your imported key material. To rotate a KMS key with imported key material, you must rotate the key manually.

Recommendations for using the AWS Encryption SDK

The AWS Encryption SDK is a powerful tool for implementing client-side encryption in your applications. Libraries are available for Java, JavaScript, C, Python, and other programming languages. It integrates with AWS Key Management Service (AWS KMS). You can also use it as a stand-alone SDK without referencing KMS keys.

Recommended practices for using this tool include carefully considering the requirements of your application. Balance those requirements against risks that can be introduced by certain configurations, such as introducing key caching into your application. For more information about data key caching, see Data key caching in the AWS Encryption SDK documentation.

Consider the following questions when determining whether to use the AWS Encryption SDK:

- Is there a requirement for client-side encryption that cannot be met by server-side encryption with services that integrate with AWS KMS?
- Can you adequately protect the keys that are used to encrypt data client-side, and how will you do that?
- Are there other, fit-for-purpose encryption libraries that might fit your use case more appropriately? Consider alternative AWS offerings, such as Amazon S3 client-side encryption and the AWS Database Encryption SDK.

Find more information about choosing the right service for your use case, see the AWS Crypto Tools Documentation.

Identity and access management best practices for AWS KMS

To use AWS Key Management Service (AWS KMS), you must have credentials that AWS can use to authenticate and authorize your requests. No AWS principal has any permissions to a KMS key unless that permission is provided explicitly and never denied. There are no implicit or automatic permissions to use or manage a KMS key. The topics in this section define security best practices to help you determine which AWS KMS access management controls you should use to secure your infrastructure.

This section discusses the following identity and access management topics:

- AWS KMS key policies and IAM policies
- Least-privilege permissions for AWS KMS
- Role-based access control for AWS KMS
- Attribute-based access control for AWS KMS
- Encryption context for AWS KMS
- Troubleshooting AWS KMS permissions

AWS KMS key policies and IAM policies

The primary way to manage access to your AWS KMS resources is with *policies*. Policies are documents that describe which principals can access which resources. Policies that are attached to an AWS Identity and Access Management (IAM) identity (users, groups of users, or roles) are called identity-based policies. IAM policies that attach to resources are called resource-based policies. AWS KMS resource policies for KMS keys are called key policies. In addition to IAM policies and AWS KMS key policies, AWS KMS supports grants. Grants provide a flexible and powerful way to delegate permissions. You can use grants to issue time-bound KMS key access to IAM principals in your AWS account or in other AWS accounts.

All KMS keys have a key policy. If you don't provide one, AWS KMS creates one for you. The default key policy that AWS KMS uses differs depending on whether you create the key using the AWS KMS console or you use the AWS KMS API. We recommend that you edit the default key policy to align with your organization's requirements for least-privilege permissions. This should also align with your strategy for using IAM policies in conjunction with key policies. For more recommendations on using IAM policies with AWS KMS, see Best practices for IAM policies in the AWS KMS documentation.

You can use the key policy to delegate authorization for an IAM principal to the identity-based policy. You can also use the key policy to refine authorization in conjunction with the identity-based policy. In either case, both the key policy and identity-based policy determine access, along with any other applicable policies that scope access, such as service control policies (SCPs), resource control policies (RCPs), or permission boundaries. If the principal is in a different account than the KMS key, essentially, only cryptographic and grant actions are supported. For more information about this cross-account scenario, see Allowing users in other accounts to use a KMS key in the AWS KMS documentation.

You must use IAM identity-based policies in combination with key policies to control access to your KMS keys. Grants can also be used in combination with these policies to control access to a KMS key. To use an identity-based policy to control access to a KMS key, the key policy must allow the account to use identity-based policies. You can either specify a key policy statement that enables IAM policies, or you can explicitly specify allowed principals in the key policy.

When writing policies, make sure that you have strong controls that restrict who can perform the following actions:

- **Update, create, and delete IAM policies and KMS key policies**
- **Attach and detach identity-based policies from users, roles, and groups**
- **Attach and detach AWS KMS key polices from KMS keys**
- **Create grants for your KMS keys –** Whether you control access to your KMS keys exclusively with key policies, or you combine key policies with IAM policies, you should restrict the ability to modify the policies. Implement an approval process for changing any existing policies. An approval process can help prevent the following:
  - **Accidental loss of IAM principal permissions –** It's possible to make changes that would prevent IAM principals from being able to manage the key or use it in cryptographic operations. In extreme scenarios, it's possible to revoke key management permissions from all users. If this happens, you need to contact AWS Support to regain access to the key.
  - **Unapproved changes to KMS key policies –** If an unauthorized user gains access to the key policy, they could modify it to delegate permissions to an unintended AWS account or principal.
  - **Unapproved changes to IAM policies –** If an unauthorized user obtains a set of credentials with permissions to manage a group's membership, they could elevate their own permissions

and make changes to your IAM policies, key policies, KMS key configuration, or other AWS resource configurations.

Carefully review the IAM roles and users that are associated with the IAM principals who are designated as your KMS key administrators. This can help prevent unauthorized deletion or changes. If you need to change the principals who have access to your KMS keys, verify that the new administrator principals are added to all required key policies. Test their permissions before you delete the previous managing principal. We strongly recommend following all IAM security best practices and using temporary credentials instead of the long-term credentials.

We recommend issuing time-bound access through grants if you don't know the names of the principals at the time that the policies are created or if the principals that require access frequently change. The grantee principal can be in the same account as the KMS key or in a different account. If the principal and KMS key are in different accounts, then you must specify an identity-based policy in addition to the grant. Grants require additional management because you must call an API to create the grant and to retire or revoke the grant when it is no longer needed.

No AWS principal, including the account root user or key creator, has any permissions to a KMS key unless they are explicitly allowed and not explicitly denied in a key policy, IAM policy, or grant. By extension, you should consider what would happen if a user gains unintended access to use a KMS key and what the impact would be. To mitigate such a risk, consider the following:

- You can maintain different KMS keys for different categories of data. This helps you separate keys and maintain more concise key policies that contain policy statements that specifically target principal access to that data category. It also means that if relevant IAM credentials are accessed unintentionally, the identity tied to that access has access only to the keys specified in the IAM policy and only if the key policy allows access to that principal.
- You can assess if a user with unintended access to the key can access the data. For example, with Amazon Simple Storage Service (Amazon S3), the user must also have the appropriate permissions to access encrypted objects in Amazon S3. Alternately, if a user has unintended access (by using RDP or SSH) to an Amazon EC2 instance that has a volume encrypted with a KMS key, the user could access the data by using operating system tools.

Note

AWS services that use AWS KMS don't expose the ciphertext to users (most current approaches to cryptoanalysis require access to the ciphertext). Additionally, ciphertext is not available for physical examination outside of an AWS data center because all storage media is physically destroyed when it is decommissioned, in accordance with NIST SP800-88 requirements.

Least-privilege permissions for AWS KMS

Because your KMS keys protect sensitive information, we recommend following the principle of least privileged access. Delegate the minimum permissions required to perform a task when you define your key policies. Allow all actions (`kms:*`) on a KMS key policy only if you plan to further restrict permissions with additional identity-based policies. If you plan to manage permissions with identity-based policies, limit who has the ability to create and attach IAM policies to IAM principals and monitor for policy changes.

If you allow all actions (`kms:*`) in both the key policy and the identity-based policy, the principal has both administrative and usage permissions to the KMS key. As a security best practice, we recommend delegating these permissions only to specific principals. Consider how you assign permissions to principals who will manage your keys and principals who will use your keys. You can do this by explicitly naming the principal in the key policy or by limiting which principals the identity-based policy is attached to. You can also use condition keys to restrict permissions. For example, you can use the aws:PrincipalTag to allow all actions if the principal making the API call has the tag specified in the condition rule.

For help understanding how policy statements are evaluated in AWS, see Policy evaluation logic in the IAM documentation. We recommend reviewing this topic before writing policies to help reduce the chance that your policy has unintended effects, such as providing access to principals that should not have access.

Tip

When testing an application in a non-production environment, use AWS Identity and Access Management Access Analyzer (IAM Access Analyzer) to help you apply least-privilege permissions in your IAM policies.

If you use IAM users instead of IAM roles, we strongly recommend using AWS multi-factor authentication (MFA) to mitigate the vulnerability of long-term credentials. You can use MFA to do the following:

- Require that users validate their credentials with MFA before performing privileged actions, such as scheduling key deletion.
- Split ownership of an administrator account password and MFA device between individuals to implement split authorization.

For sample policies that can help you configure least-privilege permissions, see IAM policy examples in the AWS KMS documentation.

Role-based access control for AWS KMS

Role-based access control (RBAC) is an authorization strategy that provides users with only the permissions required to perform their job duties, and nothing more. It is an approach that can help you implement the principle of least privilege.

AWS KMS supports RBAC. It allows you to control access to your keys by specifying granular permissions within key policies. Key policies specify a resource, action, effect, principal, and optional conditions to grant access to keys. To implement RBAC in AWS KMS, we recommend separating the permissions for key users and key administrators.

For key users, assign only the permissions that the user needs. Use the following questions to help you further refine permissions:

- Which IAM principals need access to the key?
- Which actions does each principal need to perform with the key? For example, does the principal need only `Encrypt` and `Sign` permissions?
- Which resources does the principal need to access?
- Is the entity a human or an AWS service? If it's a service, you can use the kms:ViaService condition key to restrict key usage to a specific service.

For key administrators, assign only the permissions that the administrator needs. For example, an administrator's permissions might vary depending on whether the key is used in test or production environments. If you use less restrictive permissions in certain non-production environments, implement a process to test the policies before they are released to production.

For sample policies that can help you configure role-based access control for key users and administrators, see RBAC for AWS KMS.

Attribute-based access control for AWS KMS

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. Like RBAC, it is an approach that can help you implement the principle of least privilege.

AWS KMS supports ABAC by allowing you to define permissions based on tags associated with the target resource, such as a KMS key, and tags associated with the principal making the API call. In AWS KMS, you can use tags and aliases to control access to your customer managed keys. For example, you can define IAM policies that use tag condition keys to allow operations when the principal's tag matches the tag associated with the KMS key. For a tutorial, see Define permissions to access AWS resources based on tags in the AWS KMS documentation.

As a best practice, use ABAC strategies to simplify IAM policy management. With ABAC, administrators can use tags to allow access to new resources instead of updating existing policies. ABAC requires fewer policies because you don't have to create different policies for different job functions. For more information, see Comparison of ABAC to the traditional RBAC model in the IAM documentation.

Apply the best practice of least-privilege permissions to the ABAC model. Provide IAM principals with only the permissions they need to perform their jobs. Carefully control access to tagging APIs that would allow users to modify tags on roles and resources. If you use key alias condition keys to support ABAC in AWS KMS, make sure that you also have strong controls that restrict who can create keys and modify aliases.

You can also use tags to link a specific key to a business category and verify that the correct key is being used for a given action. For example, you can use AWS CloudTrail logs to verify that the key used to perform a specific AWS KMS action belongs to the same business category as the resource that it's being used on.

⚠ **Warning**

Do not include confidential or sensitive information in the tag key or tag value. Tags are not encrypted. They are accessible to many AWS services, including billing.

Before implementing an ABAC approach to your access control, consider whether the other services that you use support this approach. For help determining which services support ABAC, see AWS services that work with IAM in the IAM documentation.

For more information about implementing ABAC for AWS KMS and the conditions keys that can help you configure policies, see ABAC for AWS KMS.

Encryption context for AWS KMS

All AWS KMS cryptographic operations with symmetric encryption KMS keys accept an encryption context. *Encryption context* is an optional set of non-secret key–value pairs that can contain additional contextual information about the data. As a best practice, you can insert encryption context in `Encrypt` operations in AWS KMS to enhance the authorization and auditability of your decryption API calls to AWS KMS. AWS KMS uses the encryption context as additional authenticated data (AAD) to support authenticated encryption. The encryption context is cryptographically bound to the ciphertext so that the same encryption context is required to decrypt the data.

The encryption context is not secret and not encrypted. It appears in plaintext in AWS CloudTrail logs so that you can use it to identify and categorize your cryptographic operations. Because the encryption context is not secret, you should allow only authorized principals to access your CloudTrail log data.

You can also use the kms:EncryptionContext:context-key and kms:EncryptionContextKeys condition keys to control access to a symmetric encryption KMS key based on the encryption context. You can also use these condition keys to require that encryption contexts are used in cryptographic operations. For these condition keys, review the guidance about the use of `ForAnyValue` or `ForAllValues` set operators in order to make sure that your policies reflect your intended permissions.

Troubleshooting AWS KMS permissions

When you write access control policies for a KMS key, consider how the IAM policy and key policy work together. The effective permissions for a principal are the permissions that are granted (and not explicitly denied) by all of the effective policies. Within an account, the permissions to a KMS key can be affected by IAM identity-based policies, key policies, permissions boundaries, service control policies, or session policies. For example, if you use both identity-based and key policies to control access to the KMS key, all policies relating to both the principal and the resource are evaluated to determine a principal's authorization to perform a given action. For more information, see Policy evaluation logic in the IAM documentation.

For detailed information and a flowchart for troubleshooting key access, see Troubleshooting key access in the AWS KMS documentation.

To troubleshoot an access denied error message

1. Confirm that the IAM identity-based policies and KMS key policies allow access.
2. Confirm that a permissions boundary in IAM is not restricting access.
3. Confirm that a service control policy (SCP) or resource control policy (RCP) in AWS Organizations is not restricting access.
4. If you are using VPC endpoints, confirm that the endpoint policies are correct.
5. In the identity-based policies and key policies, remove any conditions or resource references that restrict access to the key. After removing these restrictions, confirm that the principal can successfully call the API that previously failed. If successful, reapply the conditions and resource references one at a time and, after each, verify that the principal still has access. This helps you identify the condition or resource reference that is causing the error.

For more information, see Troubleshooting access denied error messages in the IAM documentation.

Detection and monitoring best practices for AWS KMS

Detection and monitoring are an important part of understanding the availability, state, and usage of your AWS Key Management Service (AWS KMS) keys. Monitoring helps maintain the security, reliability, availability, and performance of your AWS solutions. AWS provides several tools for monitoring your KMS keys and AWS KMS operations. This section describes how to configure and use these tools to gain greater visibility into your environment and monitor the usage of your KMS keys.

This section discusses the following detection and monitoring topics:

- Monitoring AWS KMS operations with AWS CloudTrail
- Monitoring access to KMS keys with IAM Access Analyzer
- Monitoring the encryption settings of other AWS services with AWS Config
- Monitoring KMS keys with Amazon CloudWatch alarms
- Automating responses with Amazon EventBridge

Monitoring AWS KMS operations with AWS CloudTrail

AWS KMS is integrated with AWS CloudTrail, a service that can record all calls to AWS KMS by users, roles, and other AWS services. CloudTrail captures all API calls to AWS KMS as events, including calls from the AWS KMS console, AWS KMS APIs, AWS CloudFormation, the AWS Command Line Interface (AWS CLI), and AWS Tools for PowerShell.

CloudTrail logs all AWS KMS operations, including read-only operations, such as `ListAliases` and `GetKeyRotationStatus`. It also logs operations that manage KMS keys, such as `CreateKey` and `PutKeyPolicy, and cryptographic operations, such as GenerateDataKey` and `Decrypt`. It also logs internal operations that AWS KMS calls for you, such as `DeleteExpiredKeyMaterial`, `DeleteKey`, `SynchronizeMultiRegionKey`, and `RotateKey`.

CloudTrail is enabled on your AWS account when you create it. By default, the Event history provides a viewable, searchable, downloadable, and immutable record of the past 90 days of recorded management-event API activity in an AWS Region. To monitor or audit the usage of your KMS keys beyond the 90 days, we recommend creating a CloudTrail trail for your AWS account. If you have created an organization in AWS Organizations, you can create an organization trail or an event data store that logs events for all AWS accounts in that organization.

After you establish a trail for your account or organization, you can use other AWS services to store, analyze, and automatically respond to events that are logged in the trail. For example, you can do the following:

- You can set up Amazon CloudWatch alarms that notify you of certain events in the trail. For more information, see Monitoring KMS keys with Amazon CloudWatch alarms in this guide.
- You can create Amazon EventBridge rules that automatically perform an action when an event occurs in the trail. For more information, see Automating responses with Amazon EventBridge in this guide.
- You can use Amazon Security Lake to collect and store logs from multiple AWS services, including CloudTrail. For more information, see Collecting data from AWS services in Security Lake in the Amazon Security Lake documentation.
- To enhance your analysis of operational activity, you can query CloudTrail logs with Amazon Athena. For more information, see Query AWS CloudTrail logs in the Amazon Athena documentation.

For more information about monitoring AWS KMS operations with CloudTrail, see the following:

- Logging AWS KMS API calls with AWS CloudTrail
- Examples of AWS KMS log entries
- Monitor KMS keys with Amazon EventBridge
- CloudTrail integration with Amazon EventBridge

Monitoring access to KMS keys with IAM Access Analyzer

AWS Identity and Access Management Access Analyzer (IAM Access Analyzer) helps you identify the resources in your organization and accounts (such as KMS keys) that are shared with an external entity. This service can help you identify unintended or overly broad access to your resources and data, which is a security risk. IAM Access Analyzer identifies resources that are shared with external principals by using logic-based reasoning to analyze the resource-based policies in your AWS environment.

You can use IAM Access Analyzer to identify which external entities have access to your KMS keys. When you enable IAM Access Analyzer, you create an analyzer for an entire organization or for a target account. The organization or account you choose is known as the *zone of trust* for the analyzer. The analyzer monitors the supported resources within the zone of trust. Any access to resources by principals within the zone of trust is considered trusted.

For KMS keys, IAM Access Analyzer analyzes the key policies and grants applied to a key. It generates a finding if a key policy or grant allows an external entity to access the key. Use IAM Access Analyzer to determine if external entities have access to your KMS keys, and then verify whether those entities should have access.

For more information about using IAM Access Analyzer to monitor KMS key access, see the following:

- Using AWS Identity and Access Management Access Analyzer
- IAM Access Analyzer resource types for external access
- IAM Access Analyzer resource types: AWS KMS keys
- Findings for external and unused access

Monitoring the encryption settings of other AWS services with AWS Config

AWS Config provides a detailed view of the configuration of AWS resources in your AWS account. You can use AWS Config to verify that the AWS services that use your KMS keys have their encryption settings configured appropriately. For example, you can use the encrypted-volumes AWS Config rule to validate that your Amazon Elastic Block Store (Amazon EBS) volumes are encrypted.

AWS Config includes *managed rules* that help you quickly choose rules against which to assess your resources. Check AWS Config in your AWS Regions to determine if the managed rules you need are supported in that Region. Available managed rules include checks for configuration of Amazon Relational Database Service (Amazon RDS) snapshots, CloudTrail trail encryption, default encryption for Amazon Simple Storage Service (Amazon S3) buckets, Amazon DynamoDB table encryption, and more.

You can also create *custom rules* and apply your business logic to determine whether your resources are compliant with your requirements. Open source code for many managed rules is available in the AWS Config Rules Repository on GitHub. These can be a useful starting point for developing your own custom rules.

When a resource is noncompliant with a rule, you can initiate responsive actions. AWS Config includes remediation actions that AWS Systems Manager Automation carries out. For example, if you have applied the cloud-trail-encryption-enabled rule and the rule returns a `NON_COMPLIANT` result, AWS Config can initiate an Automation document that remediates the problem by encrypting the CloudTrail logs for you.

AWS Config lets you proactively check for compliance with AWS Config rules before you provision resources. Applying rules in proactive mode helps you evaluate the configurations of your cloud resources before they are created or updated. Applying rules in proactive mode as part of your deployment pipeline lets you test resource configurations before you deploy your resources.

You can also implement AWS Config rules as controls through AWS Security Hub. Security Hub offers security standards that you can apply to your AWS accounts. These standards help you assess your environment against recommend practices. The AWS Foundational Security Best Practices standard includes controls within the protect control category to verify that encryption at rest is configured and that KMS key policies follow recommended practices.

For more information about using AWS Config to monitor the encryption settings in AWS services, see the following:

- Getting started with AWS Config
- AWS Config managed rules
- AWS Config custom rules
- Remediating noncompliant resources with AWS Config

Monitoring KMS keys with Amazon CloudWatch alarms

Amazon CloudWatch monitors your AWS resources and the applications you run on AWS in real time. You can use CloudWatch to collect and track *metrics*, which are variables that you can measure.

The expiration of imported key material, or the deletion of a key, are potentially catastrophic events if they are unintended or not properly planned for. We recommend that you configure CloudWatch alarms to alert you to these events before they occur. We also recommend that you configure AWS Identity and Access Management (IAM) policies or AWS Organizations service control policies (SCPs) to prevent the deletion of important keys.

CloudWatch alarms help you take corrective action, such as cancelling key deletion, or remediation actions, such as reimporting deleted or expired key material.

Automating responses with Amazon EventBridge

You can also use Amazon EventBridge to notify you of important events that affect your KMS keys. EventBridge is an AWS service that delivers a near real-time stream of system events that describe changes to AWS resources. EventBridge automatically receives events from CloudTrail and Security Hub. In EventBridge, you can create rules that respond to events recorded by CloudTrail.

AWS KMS events include the following:

- The key material in a KMS key was automatically rotated
- The imported key material in a KMS key expired
- A KMS key that had been scheduled for deletion was deleted

These events can initiate additional actions in your AWS account. These actions are different from the CloudWatch alarms described in the previous section because they can only be acted on after the event occurs. For example, you might want to delete resources that are connected to a specific key after that key has been deleted, or you might want to inform a compliance or auditing team that the key has been deleted.

You can also filter for any other API event that is logged in CloudTrail by using EventBridge. This means that if key policy-related API actions are of specific concern, you can filter for them. For example, you could filter in EventBridge for the `PutKeyPolicy` API action. More broadly, you can filter for any API action that starts with `Disable*` or `Delete*` to initiate automated responses.

Using EventBridge, you can monitor (which is a detective control) and investigate and respond (which are responsive controls) to unexpected or selected events. For example, you can alert security teams and take specific actions if an IAM user or role is created, when a KMS key is created, or when a key policy is changed. You can create an EventBridge event rule that filters the API actions you specify and then associate targets to the rule. Example targets include AWS Lambda functions, Amazon Simple Notification Service (Amazon SNS) notifications, Amazon Simple Queue Service (Amazon SQS) queues, and more. For more information about sending events to targets, see Event bus targets in Amazon EventBridge.

For more information about monitoring AWS KMS with EventBridge and automating responses, see Monitor KMS keys with Amazon EventBridge in the AWS KMS documentation.

Cost and billing management best practices for AWS KMS

Through breadth and depth, AWS services offer the flexibility to manage your costs while meeting business requirements. This section covers pricing for key storage in AWS Key Management Service (AWS KMS), and it provides recommendations to reduce costs, such as through key caching. You can also review KMS key usage to determine if there are additional opportunities to reduce costs.

This section discusses the following cost and billing management topics:

AWS KMS pricing for key storage

Each AWS KMS key that you create in AWS KMS incurs a charge. The monthly charge is the same for symmetric keys, asymmetric keys, HMAC keys, multi-Region keys (each primary and each replica multi-Region key), keys with imported key material, and KMS keys with a key origin of either AWS CloudHSM or an external key store.

For KMS keys that you rotate automatically or on demand, the first and second rotation of the key adds an additional monthly charge (prorated hourly) in cost. After the second rotation, any subsequent rotations in that month are not billed. Please see AWS KMS pricing for latest pricing information.

You can use AWS Budgets to configure a usage budget. AWS Budgets can alert you when the spend within your account exceeds certain thresholds. For costs related to AWS KMS, you can create a usage budget to alert based on KMS keys or requests. This can improve your visibility into your AWS KMS key storage and use costs.

Amazon S3 bucket keys with default encryption

In some use cases, workloads that access or generate large numbers of objects in Amazon Simple Storage Service (Amazon S3) can generate high volumes of requests to AWS KMS, which increases your costs. Configuring Amazon S3 bucket keys can help you reduce costs by up to 99%. This is a recommended alternative to disabling encryption to help reduce costs associated with AWS KMS.

Caching data keys by using the AWS Encryption SDK

When using the AWS Encryption SDK to perform client-side encryption, data key caching can help improve the performance of your application, reduce the risk that your application's requests to AWS KMS are throttled, and help you reduce costs. For more information about how to get started, see How to use data key caching.

Alternatives to key caching and Amazon S3 bucket keys

If key caching is not an option for you because of your data handling requirements, you can also request AWS KMS quota increases by using the AWS Management Console or the Service Quotas API. Consider the volume of API calls that you might make. The number of API calls that you make is a significant factor in AWS KMS pricing. If you increase the request-rate quota to scale your performance, the increasing number of requests to AWS KMS incurs additional costs.

Managing logging costs for KMS key usage

All AWS KMS API calls are logged to AWS CloudTrail. Applications and services can generate large volumes of AWS KMS API calls (such as for cryptographic operations, including encrypting and decrypting). It can be challenging to review CloudTrail logs without a tool that helps you organize that data, investigate trends, and search for anomalous API activity. Amazon Athena provides predefined data structures that can help you quickly set up tables for CloudTrail logs and start analyzing your log data. It is especially useful for ad-hoc analysis or further investigation during incident response. For more information, see Query AWS CloudTrail logs in the Athena documentation.

Because you pay on a per-query basis for Athena, you can set up your tables in advance at no cost. There are no charges for data definition language statements. When you are responding to an incident, this helps you make sure that many prerequisites are already met. To help you prepare, it is a best practice to write your queries after creating your table, test them, and make sure that they are producing the results you want. You can save your queries in Athena for future use. For more information about how to get started with Athena, see Getting started with Amazon Athena.

Data events provide visibility into the operations that are performed on or within a resource. These are also known as *data plane operations*. Examples include Amazon S3 `PutObject` events or Lambda function operation API calls. Data events are often high-volume activities, and you incur charges for logging them. To help control the volume of data events that are logged to trails or event data stores in CloudTrail, you can optimize your logging to reduce costs for CloudTrail, AWS KMS, and Amazon S3 by configuring advanced event selectors to limit which data events to log in CloudTrail. For more information, see How to optimize AWS CloudTrail costs by using advanced event selectors (AWS blog post).

Resources

AWS Key Management Service (AWS KMS) documentation

- AWS KMS Developer Guide
- AWS KMS API Reference
- AWS KMS in the AWS CLI Reference

Tools

- AWS Encryption SDK

AWS Prescriptive Guidance

Strategies

- Creating an encryption strategy for data at rest

Guides

- Encryption best practices and features for AWS services
- AWS Privacy Reference Architecture (AWS PRA)

Patterns

- Automatically encrypt Amazon EBS volumes
- Automatically remediate unencrypted Amazon RDS DB instances and clusters
- Monitor and remediate scheduled deletion of AWS KMS keys

Contributors

Authoring

- Frank Phillis, Senior GTM Specialist Solutions Architect, AWS
- Ken Beer, Director of AWS KMS and Crypto Libraries, AWS
- Michael Miller, Senior Solutions Architect, AWS
- Jeremy Stieglitz, Principal Product Manager, AWS
- Zach Miller, Principal Solutions Architect, AWS
- Peter M. O'Donnell, Principal Solutions Architect, AWS
- Patrick Palmer, Principal Solutions Architect, AWS
- Dave Walker, Principal Solutions Architect, AWS

Reviewing

- Manigandan Shri, Senior Delivery Consultant, AWS

Technical writing

- Lilly AbouHarb, Senior Technical Writer, AWS
- Kimberly Garmoe, Senior Technical Writer, AWS

Document history

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

Change	Description	Date
Initial publication	—	March 24, 2025

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

aggregate function

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

AI

See artificial intelligence.

AIOps

See artificial intelligence operations.

anonymization

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

anti-pattern

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

application control

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

application portfolio

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

artificial intelligence (AI)

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

artificial intelligence operations (AIOps)

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

asymmetric encryption

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

atomicity, consistency, isolation, durability (ACID)

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

attribute-based access control (ABAC)

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

authoritative data source

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

Availability Zone

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

AWS Cloud Adoption Framework (AWS CAF)

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

AWS Workload Qualification Framework (AWS WQF)

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

B

bad bot

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

BCP

See business continuity planning.

behavior graph

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

big-endian system

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

binary classification

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

bloom filter

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

blue/green deployment

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

bot

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

botnet

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

branch

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

break-glass access

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

brownfield strategy

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

buffer cache

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

business capability

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

business continuity planning (BCP)

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

C

CAF

See AWS Cloud Adoption Framework.

canary deployment

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

CCoE

See Cloud Center of Excellence.

CDC

See change data capture.

change data capture (CDC)

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

chaos engineering

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

CI/CD

See continuous integration and continuous delivery.

classification

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

client-side encryption

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

Cloud Center of Excellence (CCoE)

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

cloud computing

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

cloud operating model

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

cloud stages of adoption

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

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

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

CMDB

See configuration management database.

code repository

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

cold cache

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

cold data

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

computer vision (CV)

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

configuration drift

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

configuration management database (CMDB)

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

conformance pack

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

continuous integration and continuous delivery (CI/CD)

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

CV

See computer vision.

D

data at rest

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

data classification

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

data drift

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

data in transit

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

data mesh

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

data minimization

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

data perimeter

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

data preprocessing

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

data provenance

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

data subject

An individual whose data is being collected and processed.

data warehouse

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

database definition language (DDL)

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

database manipulation language (DML)

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

DDL

See database definition language.

deep ensemble

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

deep learning

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

defense-in-depth

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

delegated administrator

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

deployment

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

development environment

See environment.

detective control

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

development value stream mapping (DVSM)

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

digital twin

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

dimension table

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

disaster

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

disaster recovery (DR)

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

DML

See database manipulation language.

domain-driven design

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

DR

See disaster recovery.

drift detection

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

DVSM

See development value stream mapping.

E

EDA

See exploratory data analysis.

EDI

See electronic data interchange.

edge computing

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

electronic data interchange (EDI)

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

encryption

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

encryption key

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

endianness

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

endpoint

See service endpoint.

endpoint service

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

enterprise resource planning (ERP)

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

envelope encryption

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

environment

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

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

epic

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

ERP

See enterprise resource planning.

exploratory data analysis (EDA)

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

F

fact table

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

fail fast

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

fault isolation boundary

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

feature branch

See branch.

features

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

feature importance

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

feature transformation

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

few-shot prompting

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

FGAC

See fine-grained access control.

fine-grained access control (FGAC)

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

flash-cut migration

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

FM

See foundation model.

foundation model (FM)

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

G

generative AI

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

geo blocking

See geographic restrictions.

geographic restrictions (geo blocking)

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

Gitflow workflow

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

golden image

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

greenfield strategy

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

guardrail

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

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

H

HA

See high availability.

heterogeneous database migration

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

high availability (HA)

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

historian modernization

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

holdout data

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

homogeneous database migration

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

hot data

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

hotfix

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

hypercare period

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

I

IaC

See infrastructure as code.

identity-based policy

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

idle application

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

IIoT

See industrial Internet of Things.

immutable infrastructure

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

inbound (ingress) VPC

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

incremental migration

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

Industry 4.0

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

infrastructure

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

infrastructure as code (IaC)

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

industrial Internet of Things (IIoT)

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

inspection VPC

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

Internet of Things (IoT)

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

interpretability

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

IoT

See Internet of Things.

IT information library (ITIL)

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

IT service management (ITSM)

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

ITIL

See IT information library.

ITSM

See IT service management.

L

label-based access control (LBAC)

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

landing zone

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

large language model (LLM)

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

large migration

A migration of 300 or more servers.

LBAC

See label-based access control.

least privilege

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

lift and shift

See 7 Rs.

little-endian system

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

LLM

See large language model.

lower environments

See environment.

M

machine learning (ML)

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

main branch

See branch.

malware

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

managed services

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

manufacturing execution system (MES)

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

MAP

See Migration Acceleration Program.

mechanism

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

member account

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

MES

See manufacturing execution system.

Message Queuing Telemetry Transport (MQTT)

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

microservice

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

microservices architecture

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

Migration Acceleration Program (MAP)

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

migration at scale

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

migration factory

Cross-functional teams that streamline the migration of workloads through automated, agile approaches. Migration factory teams typically include operations, business analysts and owners, migration engineers, developers, and DevOps professionals working in sprints. Between 20 and 50 percent of an enterprise application portfolio consists of repeated patterns that can be optimized by a factory approach. For more information, see the discussion of migration factories and the Cloud Migration Factory guide in this content set.

migration metadata

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

migration pattern

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

Migration Portfolio Assessment (MPA)

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

Migration Readiness Assessment (MRA)

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

migration strategy

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

ML

See machine learning.

modernization

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

modernization readiness assessment

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

monolithic applications (monoliths)

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

MPA

See Migration Portfolio Assessment.

MQTT

See Message Queuing Telemetry Transport.

multiclass classification

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

mutable infrastructure

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

O

OAC

See origin access control.

OAI

See origin access identity.

OCM

See organizational change management.

offline migration

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

OI

See operations integration.

OLA

See operational-level agreement.

online migration

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

OPC-UA

See Open Process Communications - Unified Architecture.

Open Process Communications - Unified Architecture (OPC-UA)

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

operational-level agreement (OLA)

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

operational readiness review (ORR)

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

operational technology (OT)

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

operations integration (OI)

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

organization trail

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

organizational change management (OCM)

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

origin access control (OAC)

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

origin access identity (OAI)

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

ORR

See operational readiness review.

OT

See operational technology.

outbound (egress) VPC

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

P

permissions boundary

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

personally identifiable information (PII)

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

PII

See personally identifiable information.

playbook

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

PLC

See programmable logic controller.

PLM

See product lifecycle management.

policy

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

polyglot persistence

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

portfolio assessment

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

predicate

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

predicate pushdown

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

preventative control

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

principal

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

privacy by design

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

private hosted zones

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

proactive control

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

product lifecycle management (PLM)

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

production environment

See environment.

programmable logic controller (PLC)

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

prompt chaining

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

pseudonymization

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

publish/subscribe (pub/sub)

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

Q

query plan

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

query plan regression

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

R

RACI matrix

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

RAG

See Retrieval Augmented Generation.

ransomware

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

RASCI matrix

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

RCAC

See row and column access control.

read replica

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

re-architect

See 7 Rs.

recovery point objective (RPO)

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

recovery time objective (RTO)

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

refactor

See 7 Rs.

Region

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

regression

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

rehost

See 7 Rs.

release

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

relocate

See 7 Rs.

replatform

See 7 Rs.

repurchase

See 7 Rs.

resiliency

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

resource-based policy

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

responsible, accountable, consulted, informed (RACI) matrix

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

responsive control

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

retain

See 7 Rs.

retire

See 7 Rs.

Retrieval Augmented Generation (RAG)

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

rotation

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

row and column access control (RCAC)

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

RPO

See recovery point objective.

RTO

See recovery time objective.

runbook

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

S

SAML 2.0

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

SCADA

See supervisory control and data acquisition.

SCP

See service control policy.

secret

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

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

security by design

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

security control

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

security hardening

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

security information and event management (SIEM) system

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

security response automation

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

server-side encryption

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

service control policy (SCP)

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

service endpoint

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

service-level agreement (SLA)

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

service-level indicator (SLI)

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

service-level objective (SLO)

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

shared responsibility model

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

SIEM

See security information and event management system.

single point of failure (SPOF)

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

SLA

See service-level agreement.

SLI

See service-level indicator.

SLO

See service-level objective.

split-and-seed model

A pattern for scaling and accelerating modernization projects. As new features and product releases are defined, the core team splits up to create new product teams. This helps scale your organization’s capabilities and services, improves developer productivity, and supports rapid innovation. For more information, see Phased approach to modernizing applications in the AWS Cloud.

SPOF

See single point of failure.

star schema

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

strangler fig pattern

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

subnet

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

supervisory control and data acquisition (SCADA)

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

symmetric encryption

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

synthetic testing

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

system prompt

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

T

tags

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

target variable

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

task list

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

test environment

See environment.

training

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

transit gateway

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

trunk-based workflow

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

trusted access

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

tuning

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

two-pizza team

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

U

uncertainty

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

undifferentiated tasks

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

upper environments

See environment.

V

vacuuming

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

version control

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

VPC peering

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

vulnerability

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

W

warm cache

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

warm data

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

window function

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

workload

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

workstream

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

WORM

See write once, read many.

WQF

See AWS Workload Qualification Framework.

write once, read many (WORM)

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

Z

zero-day exploit

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

zero-day vulnerability

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

zero-shot prompting

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

zombie application

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