# I Về lý thuyết

## Giao thức cổng đường biên là gì? What is Border Gateway Protocol

<https://aws.amazon.com/what-is/border-gateway-protocol/#:~:text=Border%20Gateway%20Protocol%20(BGP)%20is,%2C%20devices%2C%20and%20communication%20technologies>.

Giao thức cổng đường biên (BGP) là một tập hợp các quy tắc xác định những tuyến mạng tốt nhất để truyền dữ liệu trên Internet. Internet bao gồm hàng nghìn mạng riêng, công cộng, của công ty và của chính phủ được liên kết với nhau thông qua các giao thức, thiết bị và công nghệ giao tiếp được chuẩn hóa. Khi bạn duyệt Internet, dữ liệu sẽ di chuyển qua nhiều mạng trước khi đến đích. Trách nhiệm của BGP là xem xét tất cả các đường dẫn sẵn sàng truyền dữ liệu và chọn tuyến tốt nhất. Ví dụ: Khi một người dùng ở Hoa Kỳ tải một ứng dụng có máy chủ gốc ở châu Âu, BGP sẽ giúp hoạt động giao tiếp đó diễn ra nhanh chóng và hiệu quả.

## What is jiter

<https://www.gearupbooster.com/vi/blog/what-is-jitter-in-networking.html>

Các gói dữ liệu thường được ngăn cách đều đặn trong một khoảng thời gian định trước. Khi có sự dao động hoặc "thay đổi độ trễ của gói" khi truyền một trong các gói dữ liệu này qua kết nối mạng của bạn, điều này dẫn đến hiện tượng giật hình mà bạn có thể nhận thấy trong cuộc gọi VoIP hoặc cuộc họp video dưới dạng âm thanh hoặc video chậm hoặc các biến dạng khó chịu khác.

Để đưa nó vào một định nghĩa, Jitter là độ lệch của tính chu kỳ của tín hiệu hoặc sự kiện định kỳ so với mục tiêu hoặc tần số thực của nó. Jitter cũng được sử dụng trong viễn thông để mô tả sự dao động về độ trễ của các gói cung cấp dữ liệu âm thanh hoặc video thông qua kết nối truyền thông.

Jitter thường bị nhầm lẫn với độ trễ, vì vậy hãy cùng nêu ra sự khác biệt. Jitter là sự thay đổi về lượng thời gian cần thiết để một gói dữ liệu di chuyển qua mạng. Các gói dữ liệu bị dừng trên đường đến người nhận do tắc nghẽn mạng. Chúng sẽ xuất hiện trong các khoảng thời gian ngẫu nhiên, giống như âm thanh bị rè trong cuộc gọi hoặc hiển thị dưới dạng video pixel trong cuộc gọi điện video. Mặt khác, độ trễ là thời gian cần thiết để một gói dữ liệu di chuyển qua mạng. Với độ trễ, toàn bộ quá trình truyền hoàn chỉnh, các bit và các mảnh, sẽ đến muộn hơn dự kiến.

## 3 what is IpSec ( Vpc support protocol security ‘ipSec’)

Link: <https://aws.amazon.com/vi/what-is/ipsec/#:~:text=IPSec%20l%C3%A0%20h%E1%BB%87%20th%E1%BB%91ng%20c%C3%A1c,c%C6%B0%E1%BB%9Dng%20b%E1%BA%A3o%20m%E1%BA%ADt%20giao%20th%E1%BB%A9c>.

-Amazon supports Internet Protocol Security (IPSec) VPN connections. Data transferred between your VPC and datacenter routes over an encrypted VPN connection to help maintain the confidentiality and integrity of data in transit. An internet gateway is not required to establish an AWS Site-to-Site VPN connection.

IPSec là hệ thống các quy tắc hoặc giao thức truyền thông dùng để thiết lập kết nối an toàn qua một mạng. Giao thức Internet (IP) là tiêu chuẩn phổ biến giúp xác định cách dữ liệu truyền qua Internet. IPSec bổ sung khả năng mã hóa và xác thực để tăng cường bảo mật giao thức. Ví dụ: IPSec xáo trộn dữ liệu tại nguồn và khôi phục dữ liệu bị xáo trộn tại đích của giao thức này. IPSec cũng xác thực nguồn dữ liệu.

## 4 what is dual home instance (VPC)

A Host or Virtual machine with more than one network interfaces is called dual-homed Instance. A dual-homed instance is a system fitted with two network interfaces (NICs) that sits between an untrusted network (like the Internet) and trusted network (such as a corporate network) to provide secure access.

## 5. What is proxy protocol ( Loadbanlacer)

<https://www.google.com/search?q=wwhat+is+proxy+protocol&oq=wwhat+is+proxy+protocol&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTILCAEQABgNGBMYgAQyCwgCEAAYDRgTGIAEMgoIAxAAGBMYFhgeMgoIBBAAGBMYFhgeMgoIBRAAGIAEGKIEMgoIBhAAGIAEGKIEMgcIBxAAGO8FMgcICBAAGO8F0gEIODkwNWowajeoAgCwAgA&sourceid=chrome&ie=UTF-8>

Proxy protocol is an internet protocol used to carry connection information from the source requesting the connection to the destination for which the connection was requested. Elastic Load Balancing uses proxy protocol version 1, which uses a human-readable header format.

## 6. Limit params of AWS

<https://docs.bitnami.com/aws/faq/get-started/understand-limits/#:~:text=static%20IP%20addresses.-,EC2%20Instances,up%20on%20your%20AWS%20account>.

Link detail:

<https://docs.aws.amazon.com/organizations/latest/userguide/orgs_reference_limits.html#:~:text=Limit%20increases%20can%20be%20granted,the%20default%20of%2010%20accounts>.

### Max have 20 EC2 in all account with in a region. Request more need create new request

### By default, AWS sets a limit of 5 static IP addresses per region. This includes IP addresses unassigned and currently assigned to a server.To increase IP addresses limit, [request a higher limit](http://aws.amazon.com/contact-us/eip_limit_request/) by providing information about the new limit and regions where it should be applied.

### The AWS default limit for all snapshots is 10000 snapshots per region.To increase the number of snapshots allowed, [contact AWS Support](https://aws.amazon.com/contact-us/) and request a higher limit.

* Max 10 account for org
* Maximum number of security group in a vpc : 5
* Max vpc in a region: 5
* What is the default concurrency limit for AWS Lambda functions: 1000
* maximum number of messages that can be stored in a standard queue in Amazon SQS: unlimit
* Lambda would time out after 15 minutes

## 7. What is Subject Alternative Name san?

The Subject Alternative Name (SAN) is an extension to the X. 509 specification that allows users to specify additional host names for a single SSL certificate. The use of the SAN extension is standard practice for SSL certificates, and it's on its way to replacing the use of the common name.

## 8. What is Server Name Indication (SNI) ?

Server Name Indication (SNI) is an extension to the TLS protocol that is supported by browsers and clients released after 2010. If you configure CloudFront to serve HTTPS requests using SNI, CloudFront associates your alternate domain name with an IP address for each edge location.

* Server Name Indication, often abbreviated SNI, is an extension to TLS that allows multiple hostnames to be served over HTTPS from the same IP address.

## 9. Database Clustering là gì?

Link: <https://topdev.vn/blog/database-clustering-la-gi-khi-nao-nen-su-dung/>

## 10 what is s3 transfer acceleration

<https://aws.amazon.com/vi/s3/transfer-acceleration/#:~:text=S3TA%20shortens%20the%20distance%20between,Edge%20Locations%20to%20your%20application>.

S3TA shortens the distance between client applications and AWS servers that acknowledge PUTS and GETS to Amazon S3 using our global network of hundreds of CloudFront Edge Locations. We automatically route your uploads and downloads through the closest Edge Locations to your application.

**Use Amazon S3 for hosting the web application and use Amazon CloudFront for faster distribution of content to geographically dispersed users** - Amazon S3 with CloudFront is a very powerful way of distributing static content to geographically dispersed users with low latency speeds. If you have objects that are smaller than 1GB or if the data set is less than 1GB in size, you should consider using Amazon CloudFront's PUT/POST commands for optimal performance. The given use case has data larger than 1GB and hence S3 Transfer Acceleration is a better option.

## 11 What is Amazon Elastic File System (EFS)

<https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html#:~:text=Amazon%20Elastic%20File%20System%20(Amazon,managing%20storage%20capacity%20and%20performance>.

Amazon Elastic File System (Amazon EFS) provides serverless, fully elastic file storage so that you can share file data without provisioning or managing storage capacity and performance.

**What is Amazon Elastic File System (EFS) and how is it different from Amazon Elastic Block Store (EBS)?**

 a.

EFS is a fully managed, scalable file storage for AWS Cloud services and on-premises resources, while EBS provides persistent block-level storage volumes for use with Amazon EC2 instances.

## 12 type of Storage

<https://www.futurelearn.com/info/courses/aws-cloud-technical-essentials/0/steps/287929#:~:text=AWS%20storage%20services%20are%20grouped,file%20storage%2C%20and%20object%20storage.&text=You%20may%20be%20familiar%20with,Explorer%20or%20Finder%20on%20MacOS>.

## 13 what is AWS Data Pipeline

<https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/what-is-datapipeline.html>

AWS Data Pipeline is a web service that you can use to automate the movement and transformation of data. With AWS Data Pipeline, you can define data-driven workflows, so that tasks can be dependent on the successful completion of previous tasks.

## 14 what is Amazon Kinesis Data Firehose

<https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html>

Within this overall flow, Kinesis Data Firehose covers stream ingestion, storage, processing, and destination delivery. It provides near real-time streaming extract, transform, and load (ETL) processing for any data, at any scale, and at low cost

## 15. What is the difference between a public IP address and an Elastic IP address in Amazon VPC?

A public IP address is assigned to an instance when it's launched and cannot be associated with a different instance. An Elastic IP address is a static IP address that can be associated with and disassociated from instances.

## 16. What is the difference between strongly consistent and eventually consistent read?

Strong consistency guarantees that all reads reflect the latest write, while eventual consistency may sometimes return stale data but offers cost savings by using half the capacity units

## 17. Batch Operations (dynamoDB)

Amazon DynamoDB also provides several operations designed for working with large batches of items, including BatchGetItem and BatchWriteItem. Using the BatchWriteItem action, you can perform up to 25 item creates or updates with a single operation. This allows you to minimize the overhead of each individual call when processing large numbers of items.

## 18. Amazon DynamoDB table can scale horizontally through the use of partitions

to meet the storage and performance requirements of your application. Each individual partition represents a unit of compute and storage capacity

* One single partition can hold about 10GB of data and supports a maximum of 3,000 read capacity units or 1,000 write capacity units.

1B,2D,3C,4A,5B,6A,7B,8B(A),9D,10A,11A(B),12B,13ACD,14CD,15ABC,16ABC,17A(C),18A,19BD,20CD(BC)

## 19. SQS (note)

- Each message can have up to 10 attributes.

- The maximum length of a message ID is 100 characters. (name | id is identify in message-queue)

- The maximum length of a receipt handle is 1,024 characters.

- long poll :max 20s

- delay queue max : 15 min (900s)

- visibility queue min-max: 30s-12 hours

-dead letter queue retain message in: default= 4day, max=14 days

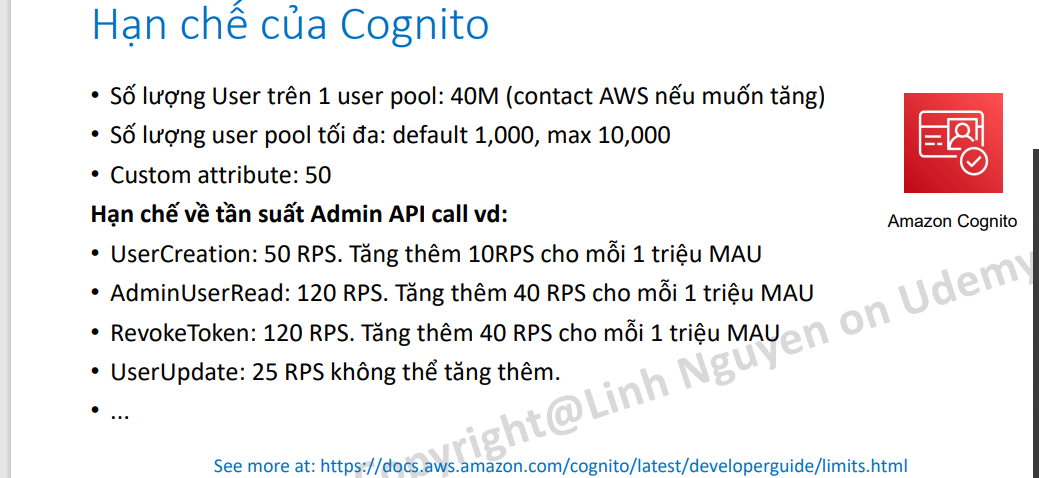
 Amazon SQS offers two options for deleting messages after processing:

* **Manual deletion** – You explicitly delete messages using the [DeleteMessage](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_DeleteMessage.html) action.
* **Automatic deletion** – Supported in certain AWS SDKs, messages are automatically deleted upon successful processing, simplifying workflows.

## 20 SNS (note)

* Số lượng topic/account 100000, FIFO: 1000
* Subcription: 12500000/topic, Fifo: 100/topic
* Delivery rate for email: 10message/s (extendable)
* Message size : 256 KB
* You can use the following protocols with Amazon SNS: HTTP, HTTPS, SMS, email, email-JSON, Amazon SQS, and AWS Lambda
* When you create a new Amazon SNS topic, an Amazon ARN is created automatically.
* Topic names should typically be available for reuse approximately 30–60 seconds after the previous topic with the same name has been deleted

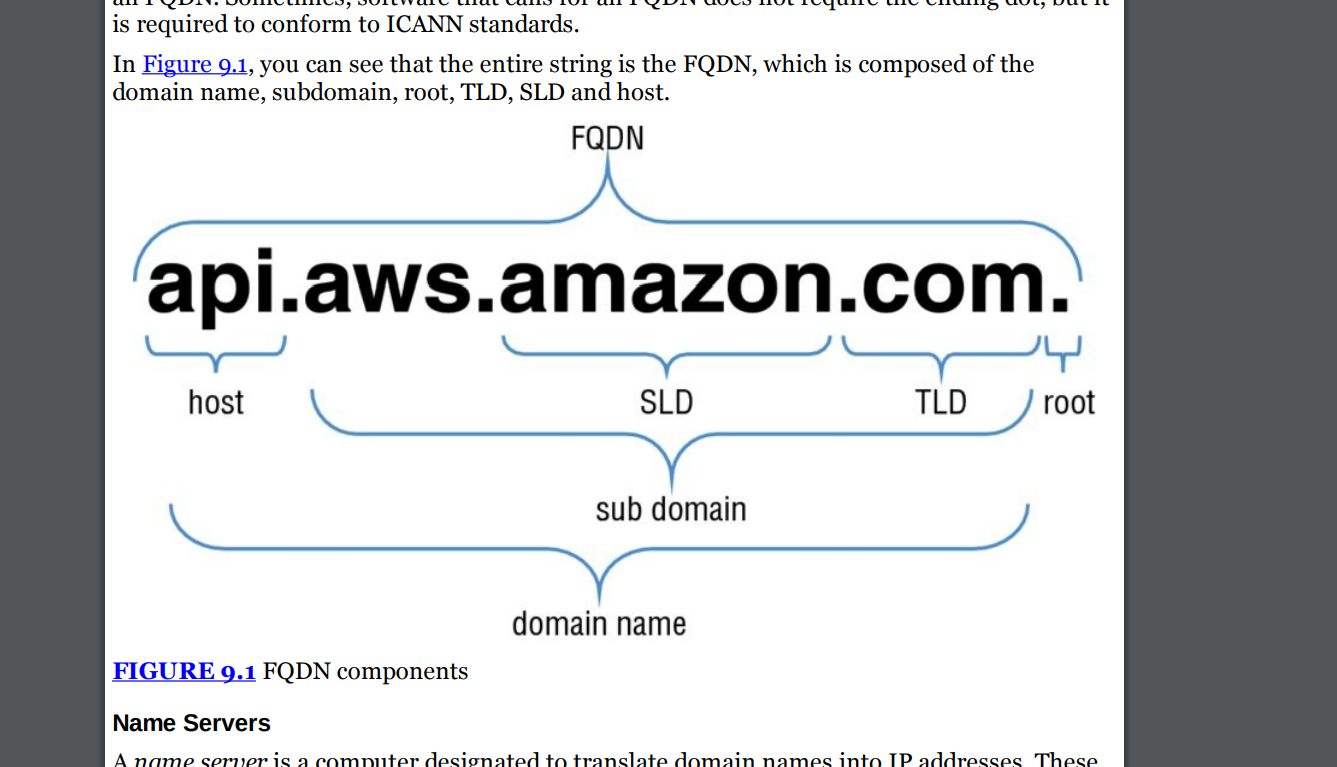
## 21 cognito (limit)



22 SWF (note)

* Actors can be workflow starters, deciders, or activity workers.
* Amazon SWF provides activity workers and deciders with work assignments, given as one of three types of tasks: activity tasks, AWS Lambda tasks, and decision tasks.
* It is possible to have more than one workflow in a domain; however, workflows in different domains cannot interact with one another.
* The decider schedules the activity tasks and provides input data to the activity workers. The decider also processes events that arrive while the workflow is in progress and closes the workflow when the objective has been completed

## 23. Khái niệm về thành phần của 1 domain (full)



* .com = TLD (top level domain)
* .amazon = SLD ( second level domain)
* .amazon.com is domain
* .Aws = sub domain
* Api = host
* Diff hostname ( chir trong1 amsy tinhs)
* Domain name (khái niệm trên internet

## 24. Cần nhớ các nhiệm vụ của record in file zone

• SOA (SOA) Record A Start of Authority (SOA) record is mandatory in all zone files, and it identifies the base DNS information about the domain. Each zone contains a single SOA record.

• A Record (Address Record): Xác định một địa chỉ IPv4 cho tên miền. Nó ánh xạ một tên miền vào một địa chỉ IP v4

• AAAA Record (IPv6 Address Record): Tương tự như A Record, nhưng sử dụng để xác định một địa chỉ IPv6 cho tên miền.

• CNAME Record (Canonical Name Record): Nó được sử dụng để tạo đường dẫn từ một tên miền thứ cấp (subdomain) đến một tên miền ở bất cứ đâu trên internet.

• MX Record (Mail Exchanger Record): Xác định các máy chủ chịu trách nhiệm nhận và xử lý thư điện tử cho một tên miền. Nó được sử dụng để định vị máy chủ email. AWS Cloud for beginner Các loại DNS Record Copyright Linh Nguyen – All right reserved 10

• TXT Record (Text Record): Cho phép bạn lưu trữ các dữ liệu văn bản tùy ý cho tên miền. Nó thường được sử dụng để xác thực tên miền và cung cấp thông tin khác nhau cho các dịch vụ khác.

• SRV Record (Service Record): Xác định vị trí và cấu hình dịch vụ cụ thể trên mạng. Nó được sử dụng chủ yếu trong việc xác định các máy chủ chịu trách nhiệm cho các dịch vụ như VoIP (Voice over IP) và IM (Instant Messaging).

• NS Record (Name Server Record): Xác định máy chủ tên miền (name server) chịu trách nhiệm quản lý các bản ghi DNS cho tên miền cụ thể. Nó cho phép bạn chỉ định máy chủ DNS mà bạn muốn sử dụng cho tên miền của mình.

• PTR Record (Pointer Record): Sử dụng để thực hiện ánh xạ địa chỉ IP thành tên miền. Nó được sử dụng chủ yếu trong việc xác định tên miền từ một địa chỉ IP cụ thể

## 25. Amazon Route 53 allows you to have several different routing policies, including the following:

* Simple—Most commonly used when you have a single resource that performs a given function for your domain
* Weighted—Used when you want to route a percentage of your traffic to one particular resource or resources
* Latency-Based—Used to route your traffic based on the lowest latency so that your users get the fastest response times
* Failover—Used for DR and to route your traffic from your resources in a primary location to a standby location
* Geolocation—Used to route your traffic based on your end user’s location

## 26 . DNS note

* uses port 53.
* DNS primarily uses UDP to serve requests.
* A. The TCP protocol is used by DNS server when the response data size exceeds 512 bytes or for tasks such as zone transfers.

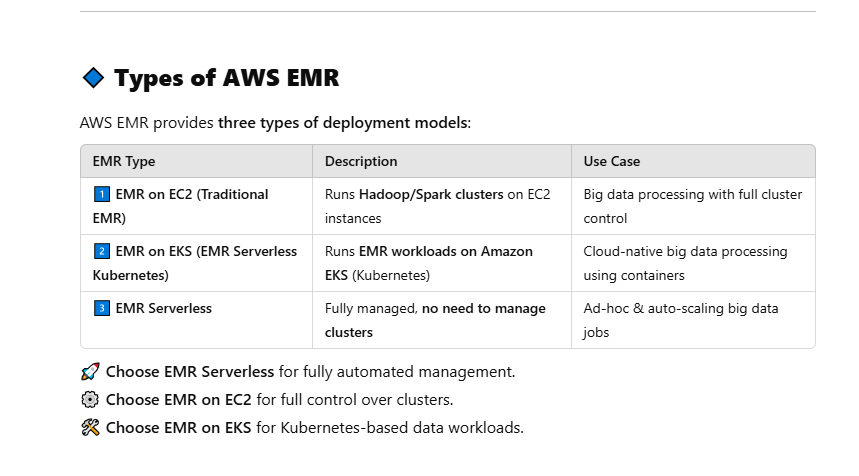
## 27 what is OpsWork

<https://docs.aws.amazon.com/opsworks-cm/latest/APIReference/Welcome.html#:~:text=AWS%20OpsWorks%20CM%20is%20a,for%20the%20servers%20to%20manage>.

AWS OpsWorks CM is a service that runs and manages configuration management servers. You can use AWS OpsWorks CM to create and manage AWS OpsWorks for Chef Automate and OpsWorks for Puppet Enterprise servers, and add or remove nodes for the servers to manage.

## 28 Amazon EMR

Link: <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-what-is-emr.html>





Amazon EMR (previously called Amazon Elastic MapReduce) is a managed cluster platform that simplifies running big data frameworks, such as [Apache Hadoop](https://aws.amazon.com/elasticmapreduce/details/hadoop) and [Apache Spark](https://aws.amazon.com/elasticmapreduce/details/spark), on AWS to process and analyze vast amounts of data. Using these frameworks and related open-source projects, you can process data for analytics purposes and business intelligence workloads. Amazon EMR also lets you transform and move large amounts of data into and out of other AWS data stores and databases, such as Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB.

* Hadoop Distributed File System (HDFS) HDFS is the standard file system that comes with Hadoop. All data is replicated across multiple instances to ensure durability. Amazon EMR can use Amazon EC2 instance storage or Amazon EBS for HDFS. When a cluster is shut down, instance storage is lost and the data does not persist. HDFS can also make use of Amazon EBS storage, trading in the cost effectiveness of instance storage for the ability to shut down a cluster without losing data.
* EMR File System (EMRFS) EMRFS is an implementation of HDFS that allows clusters to store data on Amazon S3. EMRFS allows you to get the durability and low cost of Amazon S3 while preserving your data even if the cluster is shut down. A key factor driving the type of storage a cluster uses is whether the cluster is persistent or transient. A persistent cluster continues to run 24×7 after it is launched.

2 type:

- Hadoop Distributed File System (HDFS) HDFS is the standard file system that comes with Hadoop. All data is replicated across multiple instances to ensure durability. Amazon EMR can use Amazon EC2 instance storage or Amazon EBS for HDFS.

- EMR File System (EMRFS) EMRFS is an implementation of HDFS that allows clusters to store data on Amazon S3. EMRFS allows you to get the durability and low cost of Amazon S3 while preserving your data even if the cluster is shut down.

-- note

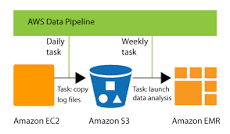
- Know what service Amazon EMR provides. Amazon EMR provides a managed Hadoop service on AWS that allows you to spin up large Hadoop clusters in minutes.

- Know the difference between persistent and transient clusters. Persistent clusters run continuously, so they do not lose data stored on instance-based HDFS. Transient clusters are launched for a specific task, then terminated, so they access data on Amazon S3 via EMRFS.

- Know the use cases for Amazon EMR. Amazon EMR is useful for big data analytics in virtually any industry, including, but not limited to, log processing, clickstream analysis, and genomics and life sciences

-

## 29 What is a data pipeline in AWS?



AWS Data Pipeline is a web service that you can use to automate the movement and transformation of data. With AWS Data Pipeline, you can define data-driven workflows, so that tasks can be dependent on the successful completion of previous tasks.

## 30 Amazon elastic cache

Cache Engines

* Memcache

+ A single Memcached cluster can contain up to 20 nodes

* Redis :
* + no support cluster but support multi AZ a up to 5 replica

+ max 5 replica

+ Cache clusters running Redis support the concept of replication groups. A replication group consists of up to six clusters, with five of them designated as read replicas

## 31 AWS CloudTrail

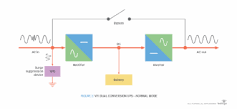
* A Trail That Applies to All Regions

When you create a trail that applies to all AWS regions, AWS CloudTrail creates the same trail in each region, records the log files in each region, and delivers the log files to the single Amazon S3 bucket (and optionally to the Amazon CloudWatch Logs log group) that you specify

* A Trail That Applies to One Region :

You specify a bucket that receives events only from that region. The bucket can be in any region that you specify. If you create additional individual trails that apply to specific regions, you can have those trails deliver event logs to a single Amazon S3 bucket

## 32 What is UPS used for?



An uninterruptible power supply (UPS) is a device that allows a computer to keep running for at least a short time when incoming power is interrupted. Provided utility power is flowing, it also replenishes and maintains energy storage.

## 33 (SnowBall) – import/export

### AWS Snowball AWS

AWS Snowball AWS Snowball uses Amazon-provided shippable storage appliances shipped through UPS. Each AWS Snowball is protected by AWS KMS and made physically rugged to secure and protect your data while the device is in transit. At the time of this writing, AWS Snowballs come in two sizes: 50TB and 80TB, and the availability of each varies by region. AWS Snowball provides the following features: - + You can import and export data between your on-premises data storage locations and Amazon S3.

+ Encryption is enforced, protecting your data at rest and in physical transit.

+ You don’t have to buy or maintain your own hardware devices.

+ You can manage your jobs through the AWS Snowball console.

+ The AWS Snowball is its own shipping container, and the shipping label is an E Ink display that automatically shows the correct address when the AWS Snowball is ready to ship. You can drop it off with UPS, no box required.

### AWS Snowball comes in two main types:

#### AWS Snowball Edge Storage Optimized

* + Best for **large-scale data migrations, backups, and analytics**.
  + Provides **80 TB of usable storage**.
  + Supports **block and Amazon S3-compatible object storage**.
  + Optional **compute capabilities** for processing data before transferring.

#### AWS Snowball Edge Compute Optimized

* + Best for **real-time data processing, analytics, and IoT applications**.
  + Provides **42 TB of usable storage**.
  + Includes **more compute power (CPU & GPU options)** for **machine learning and edge computing**.
  + Supports **AWS IoT Greengrass, EC2 instances, and Lambda functions**.

#### **AWS Snowmobile (Exabyte-Scale Transfer)**

🔹 **Best for:** **Massive data center migrations (up to 100 PB per unit)**.  
🔹 **Features:**

* **45-foot shipping container** with high-speed data transfer.
* **Tamper-resistant & military-grade security**.
* **Secure transportation via trucks to AWS data centers**.
* **Up to 100 PB per Snowmobile** – ideal for **huge data sets**.

🛠 **Use Cases:**

* Moving **entire data centers** to AWS.
* Transferring **huge video archives or genomic data**.
* **Disaster recovery & backup solutions** at scale.

#### **3. AWS Snowcone (Smallest & Most Portable)**

🔹 **Best for:** **Ultra-portable data transfer & lightweight edge computing**.  
🔹 **Features:**

* **8 TB usable storage** (SSD-based).
* **Battery-powered option available**.
* Can **run EC2 instances and AWS IoT Greengrass**.
* **Weighs only ~4.5 lbs (2.1 kg)** – fits in a backpack.

🛠 **Use Cases:**

* **Field data collection** (e.g., disaster response, remote research).
* **IoT applications & edge processing** in mobile environments.
* **Small-scale data migrations** where Snowball is too large.

### AWS Import/Export Disk

AWS Import/Export Disk supports transfers data directly onto and off of storage devices you own using the Amazon high-speed internal network.

Important things to understand about AWS Import/Export Disk include:

+ You can import your data into Amazon Glacier and Amazon EBS, in addition to Amazon S3.

+ You can export data from Amazon S3.

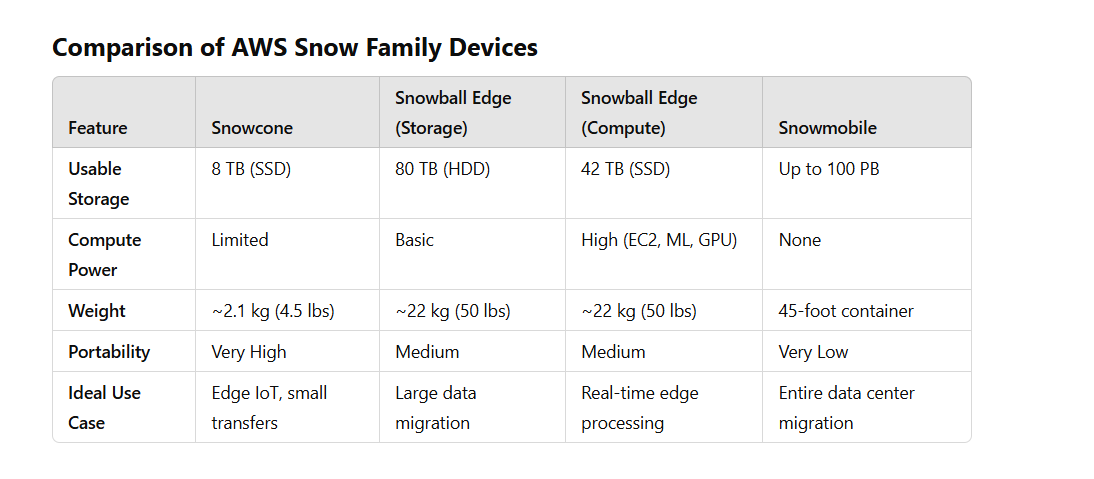
+ Encryption is optional and not enforced.

+ You buy and maintain your own hardware devices.

+ You can’t manage your jobs through the AWS Snowball console.

+ Unlike AWS Snowball, AWS Import/Export Disk has an upper limit of 16TB.

## \*\*All type SnowBall



## 34 AWS Directory Service

AWS Directory Service is a managed service offering that provides directories that contain information about your organization, including users, groups, computers, and other resources.

* AWS Directory Service for Microsoft Active Directory (Enterprise Edition) This Directory Service is your best choice if you have more than 5,000 users and need a trust relationship set up between an AWS-hosted directory and your on-premises directories.
* Simple AD In most cases, Simple AD is the least expensive option and your best choice if you have 5,000 or fewer users and don’t need the more advanced Microsoft Active Directory features. (Note that you cannot set up trust relationships between Simple AD and other Active Directory domains.)
* AD Connector AD Connector is your best choice when you want to use your existing onpremises directory with AWS cloud service

## 34.1 Detail

**AWS Managed Microsoft AD** - AWS Directory Service lets you run Microsoft Active Directory (AD) as a managed service. AWS Directory Service for Microsoft Active Directory, also referred to as AWS Managed Microsoft AD, is powered by Windows Server 2012 R2. When you select and launch this directory type, it is created as a highly available pair of domain controllers connected to your virtual private cloud (VPC).

With AWS Managed Microsoft AD, you can run directory-aware workloads in the AWS Cloud, including Microsoft SharePoint and custom .NET and SQL Server-based applications. You can also configure a trust relationship between AWS Managed Microsoft AD in the AWS Cloud and your existing on-premises Microsoft Active Directory, providing users and groups with access to resources in either domain, using single sign-on (SSO).

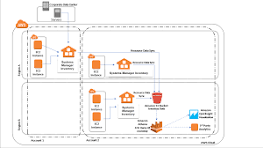
AWS Managed Microsoft AD is your best choice if you need actual Active Directory features to support AWS applications or Windows workloads, including Amazon Relational Database Service for Microsoft SQL Server. It's also best if you want a standalone AD in the AWS Cloud that supports Office 365 or you need an LDAP directory to support your Linux applications.

Incorrect options:

**AD Connector** - AD Connector is a directory gateway with which you can redirect directory requests to your on-premises Microsoft Active Directory without caching any information in the cloud. AD Connector is your best choice when you want to use your existing on-premises directory with compatible AWS services.

**Simple AD** - Simple AD is a standalone directory in the cloud, where you create and manage user identities and manage access to applications. Simple AD provides a subset of the features offered by AWS Managed Microsoft AD. However, note that Simple AD does not support features such as multi-factor authentication (MFA), trust relationships with other domains, Active Directory Administrative Center, PowerShell support, Active Directory recycle bin, group managed service accounts, and schema extensions for POSIX and Microsoft applications.

## 35 What is inventory in AWS?



Link <https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-inventory.html>

AWS Systems Manager Inventory provides visibility into your Amazon EC2 and on-premises computing environment. You can use Inventory to collect metadata from your managed instances.

## 36 Amazon CloudFront?

* What origin servers are supported by Amazon CloudFront
* Amazon CloudFront can use an Amazon S3 bucket or any HTTP server

### Whole Website

Using cache behaviors and multiple origins, you can easily use Amazon CloudFront to serve your whole website and to support different behaviors for different client devices.

### Private Content

* Signed URLs Use URLs that are valid only between certain times and optionally from certain IP addresses.
* Signed Cookies Require authentication via public and private key pairs.
* Origin Access Identities (OAI) Restrict access to an Amazon S3 bucket only to a special Amazon CloudFront user associated with your distribution. This is the easiest way to ensure that content in a bucket is only accessed by Amazon CloudFront.

##### **What is the HTTP response status code for a request that is blocked by AWS CloudFront?**

: 403

## 37 AWS Storage Gateway

**Types of AWS Storage Gateway**

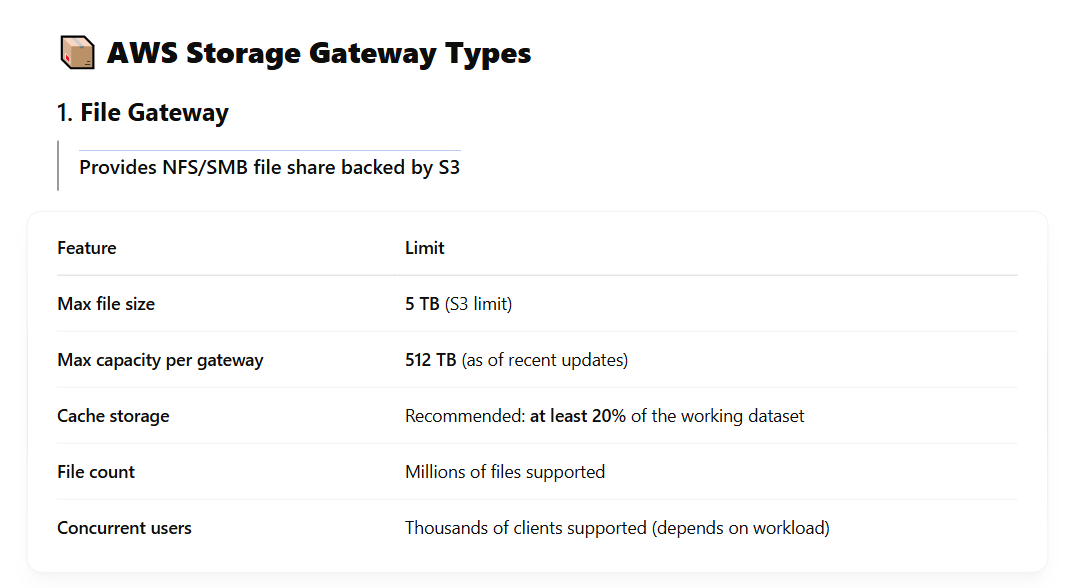
AWS Storage Gateway offers three primary types, each catering to different storage needs:

1. **File Gateway**: This gateway allows you to store files locally using standard protocols like NFS (for Linux) or SMB (for Windows). Files are stored as objects in Amazon S3, enabling seamless integration between your on-premises environment and the cloud.

[edureka.co](https://www.edureka.co/blog/aws-storage-gateway/?utm_source=chatgpt.com)

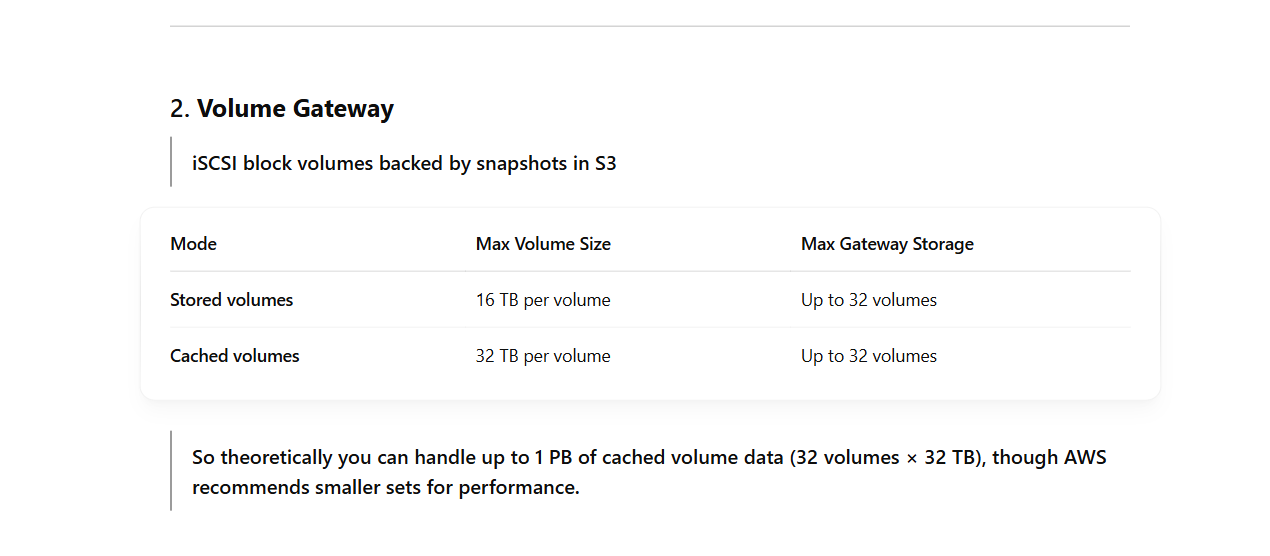
### **File Gateway**

Provides NFS/SMB file share backed by S3

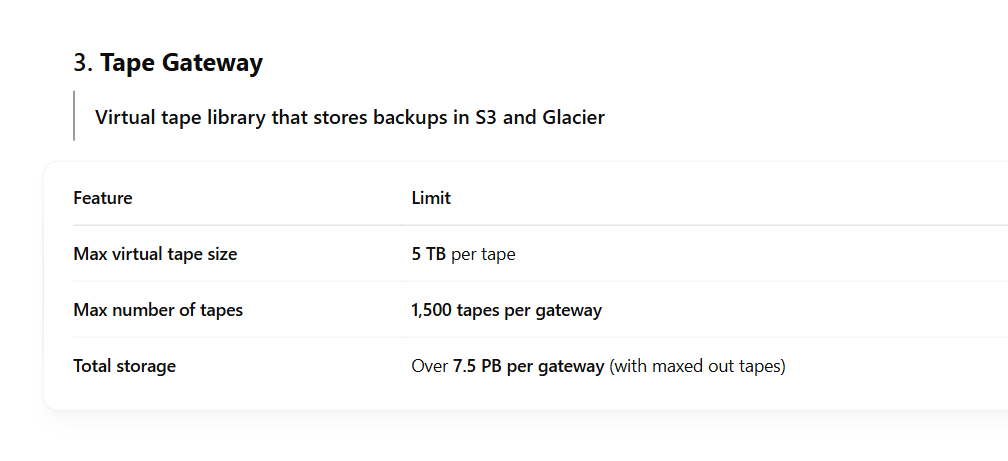


1. **Volume Gateway**: Provides block storage to your on-premises applications using the iSCSI protocol. Data on the volumes is stored in Amazon S3, and you can take point-in-time snapshots stored as Amazon EBS snapshots.

[Amazon Web Services](https://aws.amazon.com/storagegateway/faqs/?utm_source=chatgpt.com)



1. **Tape Gateway**: Offers a virtual tape library (VTL) interface, allowing you to use your existing backup applications to store data on virtual tapes. These tapes are stored in Amazon S3 and can be archived to Amazon S3 Glacier or Amazon S3 Glacier Deep Archive for long-term storage.



AWS Storage Gateway is a service connecting an on-premises software appliance with cloud- based storage to provide seamless and secure integration between an organization’s onpremises IT environment and AWS storage infrastructure. The service enables you to store data securely on the AWS cloud in a scalable and cost-effective manner

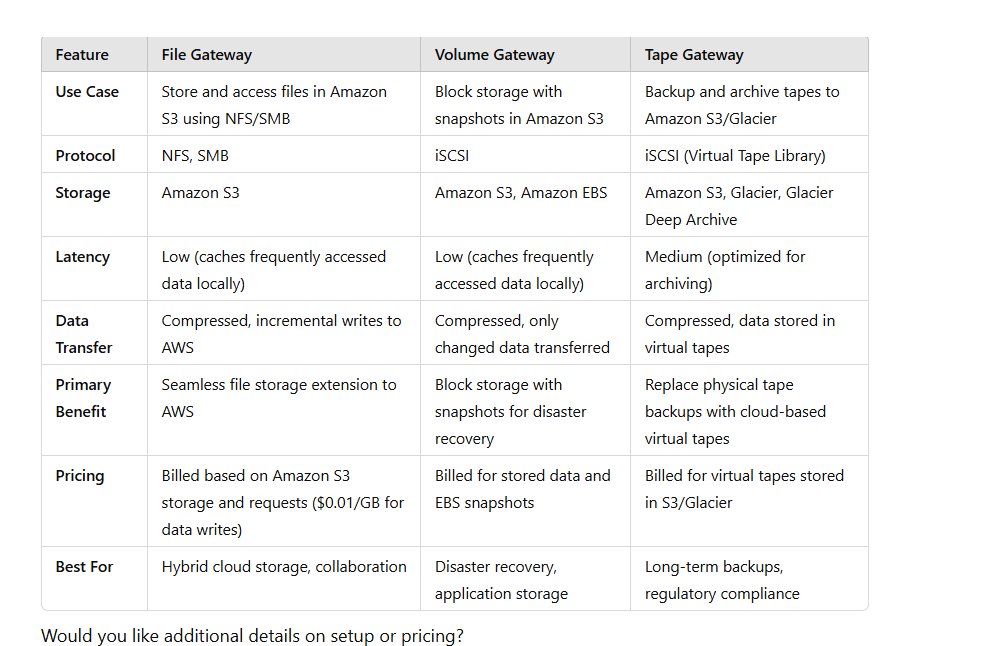
The Volume Gateway provides block storage to your on-premises applications using iSCSI connectivity. Volume Gateway store block not File

### - Gateway-Cached Volumes

Gateway-Cached volumes allow you to expand your local storage capacity into Amazon S3. All data stored on a Gateway-Cached volume is moved to Amazon S3, while recently read data is retained in local storage to provide low-latency access. While each volume is limited to a maximum size of 32TB, a single gateway can support up to 32 volumes for a maximum storage of 1 PB

### Gateway-Stored Volumes

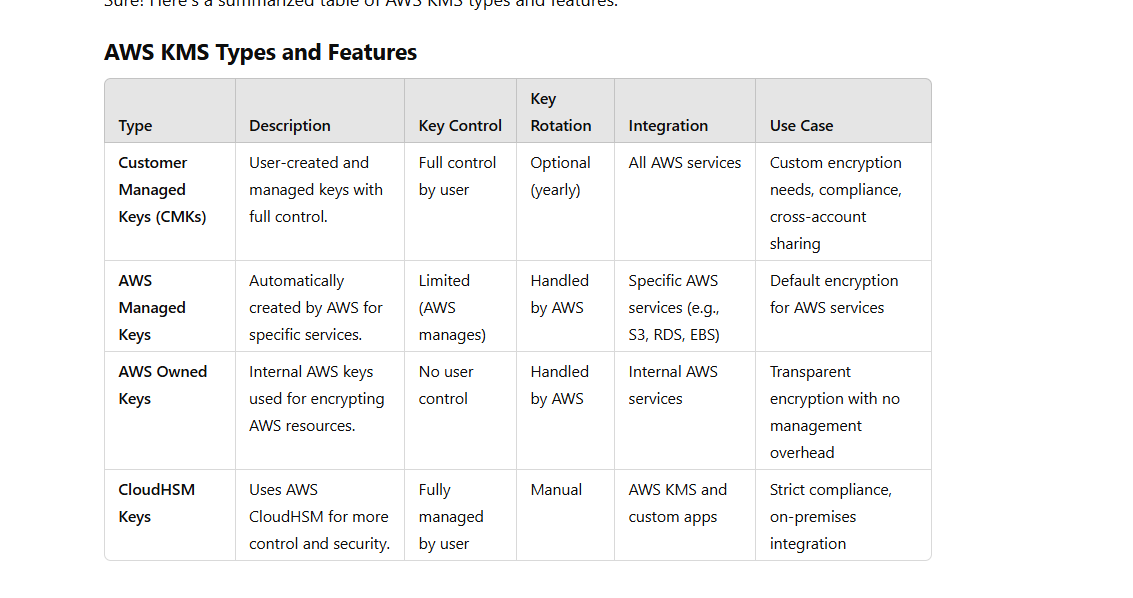
Gateway-Stored volumes allow you to store your data on your on-premises storage and asynchronously back up that data to Amazon S3. This provides lowlatency access to all data, while also providing off-site backups taking advantage of the durability of Amazon S3. The data is backed up in the form of Amazon Elastic Block Store (Amazon EBS) snapshots. While each volume is limited to a maximum size of 16TB, a single gateway can support up to 32 volumes for a maximum storage of 512TB.



## 38 AWS Key Management Service (KMS)

AWS Key Management Service (AWS KMS) AWS KMS is a managed service that makes it easy for you to create and control the encryption keys used to encrypt your data.

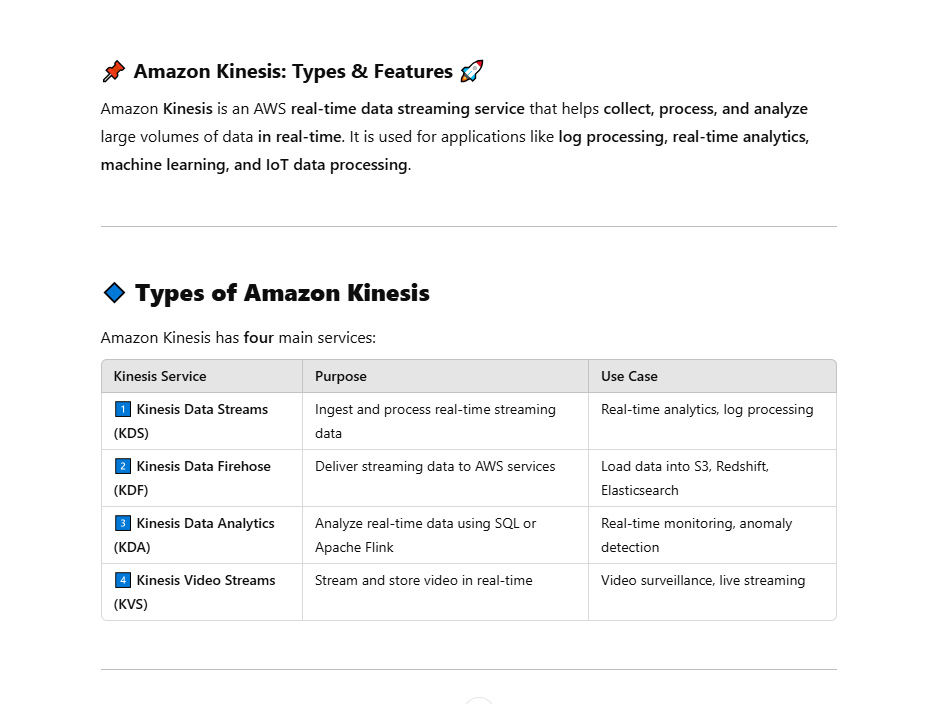
* Customer Managed Keys AWS KMS uses a type of key called a Customer Master Key (CMK) to encrypt and decrypt data. CMKs are the fundamental resources that AWS KMS manages. They can be used inside of AWS KMS to encrypt or decrypt up to 4 KB of data directly. They can also be used to encrypt generated data keys that are then used to encrypt or decrypt larger amounts of data outside of the service. CMKs can never leave AWS KMS unencrypted, but data keys can leave the service unencrypted.
* Data Keys You use data keys to encrypt large data objects within your own application outside AWS KMS. When you call GenerateDataKey, AWS KMS returns a plaintext version of the key and ciphertext that contains the key encrypted under the specified CMK. AWS KMS tracks which CMK was used to encrypt the data key. You use the plaintext data key in your application to encrypt data, and you typically store the encrypted key alongside your encrypted data. Security best practices suggest that you should remove the plaintext key from memory as soon as is practical after use. To decrypt data in your application, pass the encrypted data key to the Decrypt function. AWS KMS uses the associated CMK to decrypt and retrieve your plaintext data key. Use the plaintext key to decrypt your data, and then remove the key from memory.
* Envelope Encryption AWS KMS uses envelope encryption to protect data. AWS KMS creates a data key, encrypts it under a CMK, and returns plaintext and encrypted versions of the data key to you. You use the plaintext key to encrypt data and store the encrypted key alongside the encrypted data. The key should be removed from memory as soon as is practical after use. You can retrieve a plaintext data key only if you have the encrypted data key and you have permission to use the corresponding master key.
* Encryption Context All AWS KMS cryptographic operations accept an optional key/value map of additional contextual information called an encryption context. The specified context must be the same for both the encrypt and decrypt operations or decryption will not succeed. The encryption context is logged, can be used for additional auditing, and is available as context in the AWS policy language for fine-grained policybased authorization.



## 39 Cloud Trail

AWS CloudTrail captures AWS API calls and related events made by or on behalf of an AWS account and delivers log files to an Amazon S3 bucket that you specify

## 40 Amazon Kinesis



* Kinesis data analytic only analytic when have data from kinesis data firehoue or kineis stream. Summary kinesis cant get data from source

Amazon Kinesis is a platform for handling massive streaming data on AWS, offering powerful services to make it easy to load and analyze streaming data and also providing the ability for you to build custom streaming data applications for specialized needs.

Overview Amazon Kinesis is a streaming data platform consisting of three services addressing different real-time streaming data challenges:

* Amazon Kinesis Firehose: A service enabling you to load massive volumes of streaming data into AWS (can store to Amazon S3, Amazon Redshift, or Amazon Elasticsearch.)
* Amazon Kinesis Streams: A service enabling you to build custom applications for more complex analysis of streaming data in real time

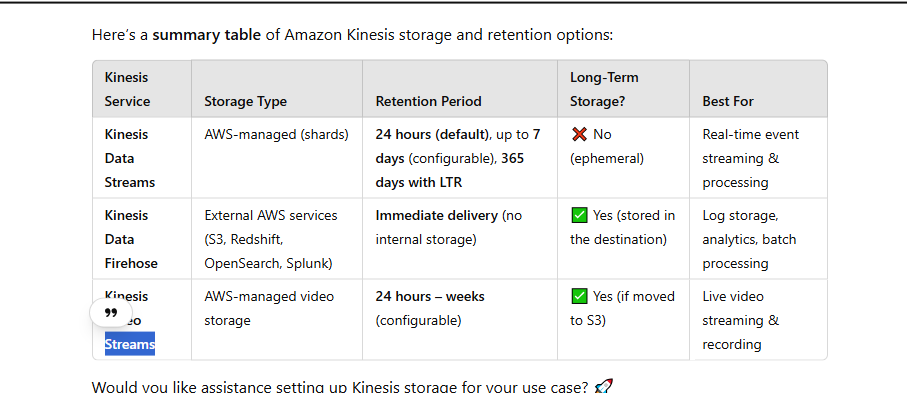
. Amazon Kinesis Streams can scale to support nearly limitless data streams by distributing incoming data across a number of shards

* Amazon Kinesis Analytics: A service enabling you to easily analyze streaming data real time with standard SQL
* **Kinesis Video Streams (KVS)**

🔸 **Purpose:** Capture, process, and analyze **real-time video streams**.  
🔸 **Use Case:** **Live video streaming, surveillance, IoT cameras, AI/ML applications**.  
🔸 **Key Features:**

* **Supports multiple formats**: H.264, WebRTC.
* **Low latency streaming**: Stream to AWS AI services for analysis.
* **Integration with AI/ML**: Works with Amazon Rekognition for **video analysis**.

### Summary storage



## 42 S3 Inventory

Link: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/storage-inventory.html>

You can use Amazon S3 Inventory to help manage your storage. For example, you can use it to audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs. You can also simplify and speed up business workflows and big data jobs by using Amazon S3 Inventory, which provides a scheduled alternative to the Amazon S3 synchronous List API operations. Amazon S3 Inventory does not use the List API operations to audit your objects and does not affect the request rate of your bucket.

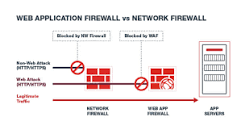
You can query Amazon S3 Inventory with standard SQL queries by using [Amazon Athena](https://docs.aws.amazon.com/athena/latest/ug/what-is.html), [Amazon Redshift Spectrum](https://docs.aws.amazon.com/redshift/latest/dg/c-getting-started-using-spectrum.html), and other tools, such as [Presto](https://prestodb.io/), [Apache Hive](https://hive.apache.org/), and [Apache Spark](https://databricks.com/spark/about/). For more information about using Athena to query your inventory files

## 43 What is the use of Amazon Athena?

Link: <https://docs.aws.amazon.com/athena/latest/ug/what-is.html>

Amazon Athena is a service that enables data analysts to perform interactive queries in the web-based cloud storage service, Amazon Simple Storage Service (S3). Athena is used with large-scale data sets. Amazon S3 is designed for online backup and archiving of data and applications on Amazon Web Services (AWS).

## 44 What is the difference between AWS firewall and WAF?



A WAF is essential when you need specialized protection for web applications against web-specific threats. Firewalls provide broader network security but may not catch application-layer attacks.

## 45 what is  AWS Global Accelerator

AWS Global Accelerator is a networking service that improves the availability and performance of applications with global users. It provides a single, static IP address that acts as a fixed entry point to route traffic to the optimal AWS endpoint, such as EC2 instances, ALBs, or NLBs, across multiple AWS Regions.

### **Key Features**

1. **Static Anycast IPs** – Provides two static IP addresses that remain the same regardless of the backend changes.
2. **Global Traffic Routing** – Routes user requests to the closest healthy AWS endpoint using the AWS global network.
3. **Automatic Failover** – If a failure occurs, traffic is redirected to the next best available endpoint.
4. **Health Checks** – Continuously monitors the health of application endpoints.
5. **DDoS Protection** – Integrated with AWS Shield for security against distributed denial-of-service (DDoS) attacks.
6. **Improved Latency** – Uses the AWS global backbone network to reduce latency compared to standard internet routing.

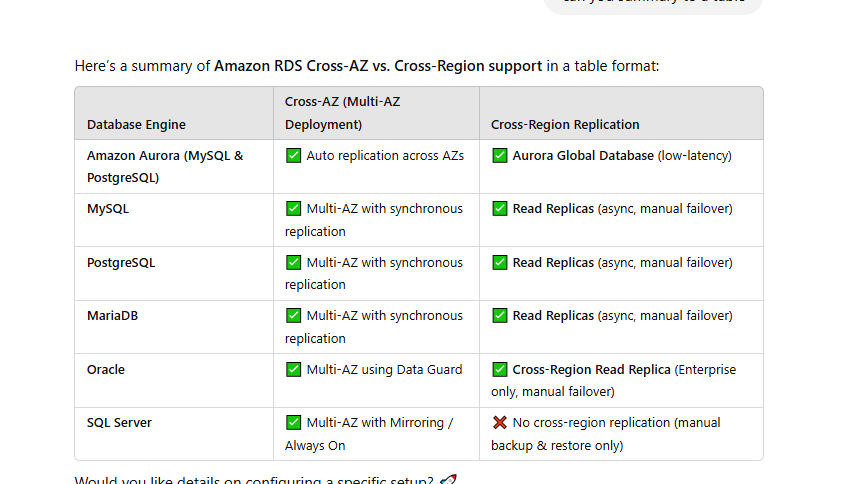
### **Use Cases**

## 46 What is the purpose of VPC Flow Logs?

## 47 RDS

Which Amazon RDS engine supports encryption at rest as a default option?

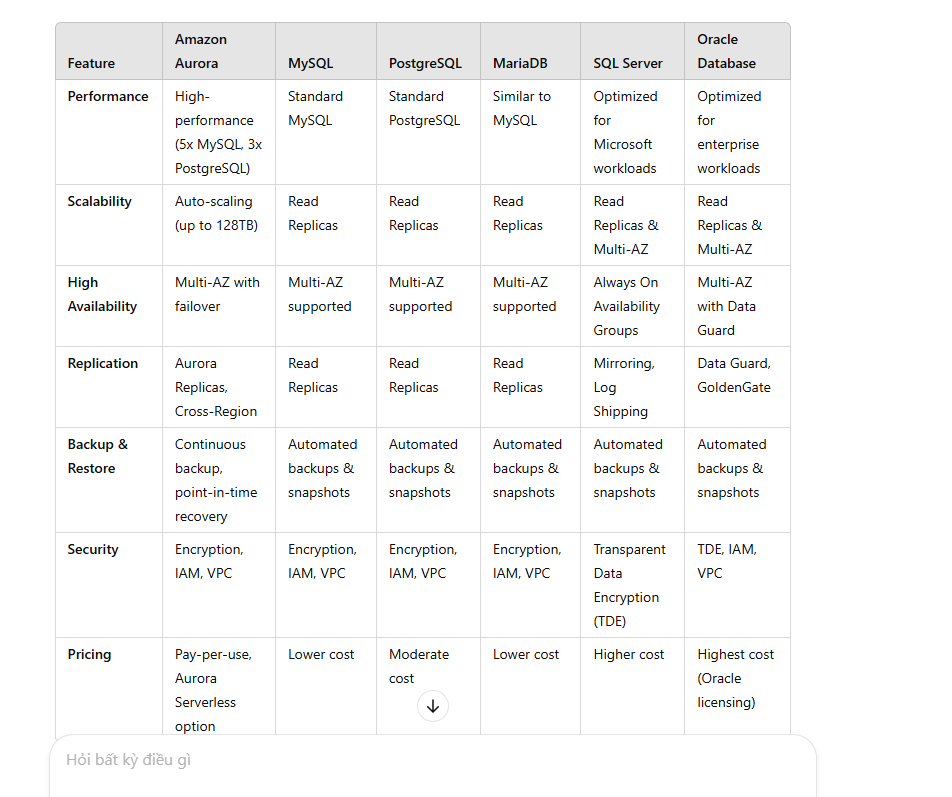
No is optional

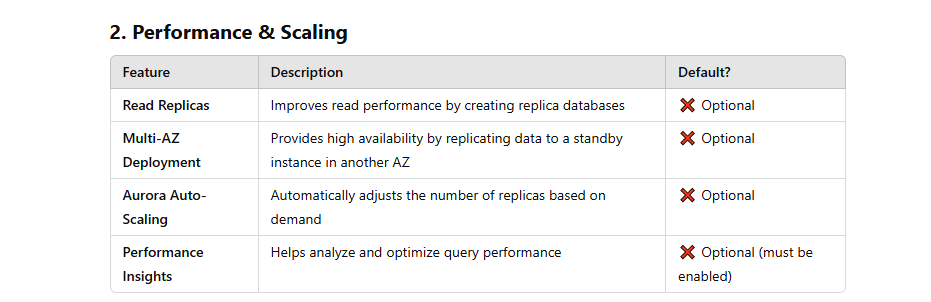


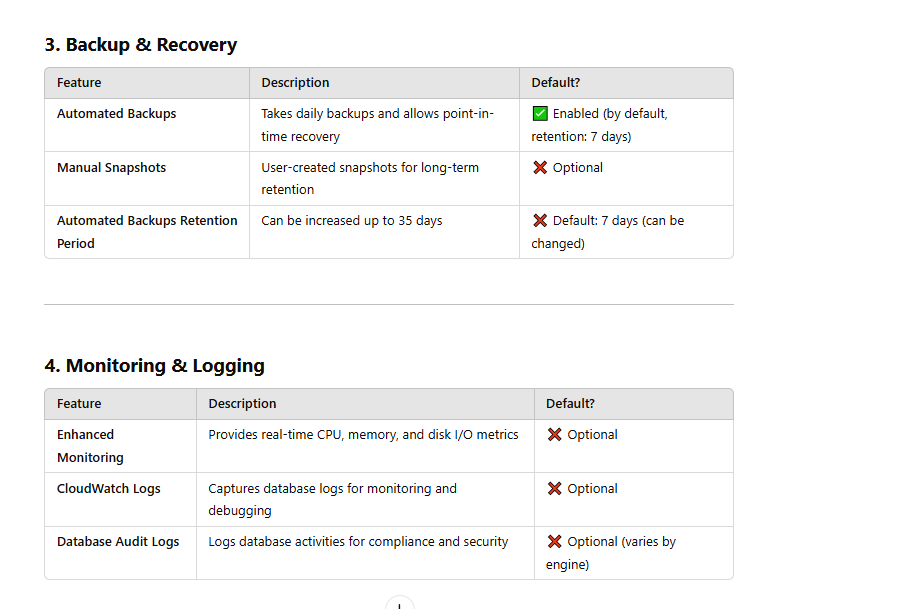
 **MySQL, MariaDB, PostgreSQL, and Oracle** support **cross-region read replicas**.

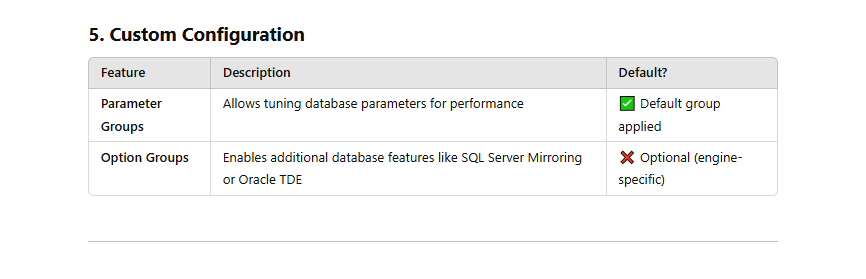
 You can set up **read replicas** in different AWS Regions, but **failover is manual** (unlike Aurora Global Database).

Aurora global cross region auto failover



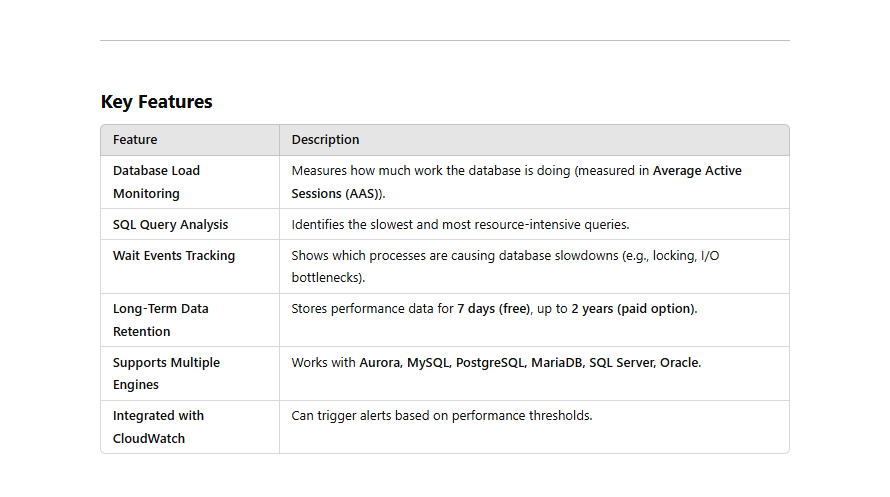






### **What is Amazon RDS Performance Insights?**

**Performance Insights** is an **optional** performance monitoring feature in Amazon RDS that helps analyze and troubleshoot database load. It provides **real-time metrics** and **historical data** to identify performance bottlenecks.



## 48 Amazon CloudWatch

CloudWatch is the AWS service designed to monitor and troubleshoot applications running across hybrid environments. It provides a unified view of operational health and collects metrics, logs, and events from various AWS resources, on-premises servers, and applications

 CloudWatch enables you to set alarms, visualize logs and metrics side by side, take automated actions, and gain system-wide visibility into resource utilization, application performance, and operational health. This makes it an ideal tool for monitoring and troubleshooting applications in hybrid environments, offering insights that span both cloud and on-premises infrastructure.

## 49 \* Access instance metadata for an EC2 instance

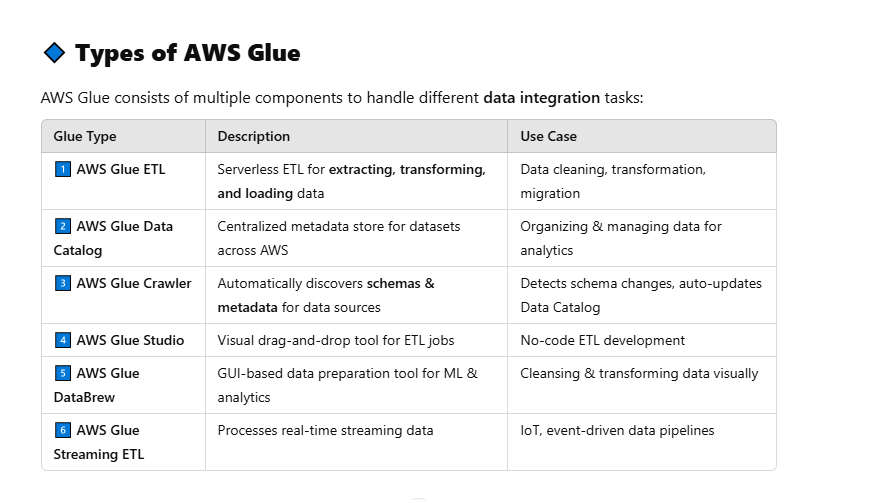
Link: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instancedata-data-retrieval.html>

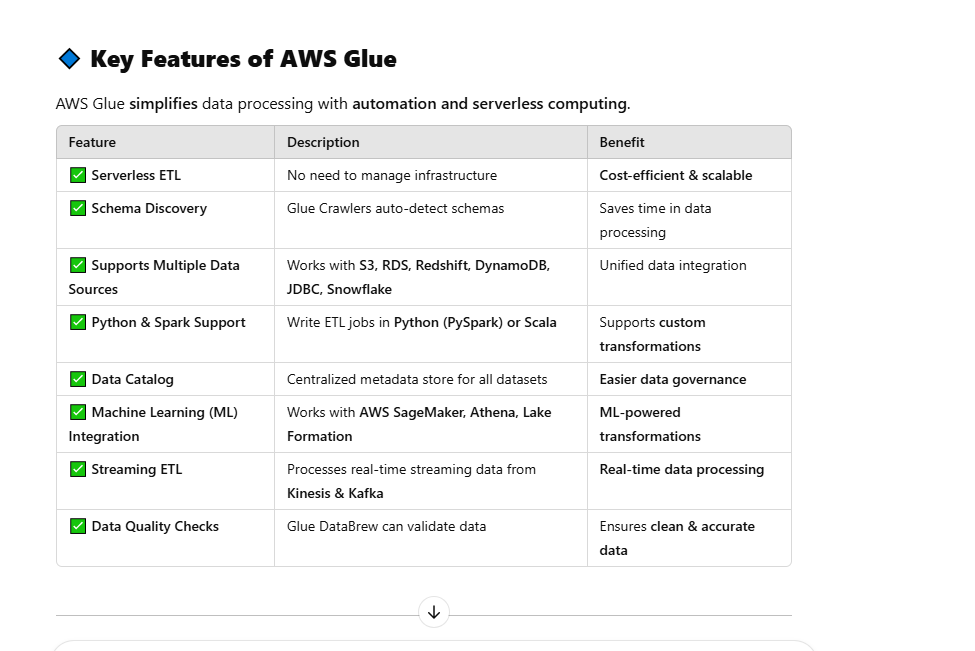
The correct answer, 'curl http://169.254.169.254/latest/user-data', allows you to view the user data script directly from the instance without accessing the log file.

## 50 what is  AWS Glue?

AWS Glue is a fully managed ETL service that makes it easy for customers to prepare and load their data for analytics. You can create and run an ETL job with a few clicks in the AWS Management Console .

AWS **Glue** is a **serverless data integration service** that helps **extract, transform, and load (ETL) data** from multiple sources. It automates **data discovery, transformation, and cataloging** for **data lakes, analytics, and machine learning (ML) workflows**.





Link: <https://docs.aws.amazon.com/glue/latest/dg/add-classifier.html>

* A classifier is a metadata table used to infer schema of data

## 51 What is the difference between a public IP address and an Elastic IP address in Amazon VPC?

A public IP address is assigned to an instance when it's launched and cannot be associated with a different instance. An Elastic IP address is a static IP address that can be associated with and disassociated from instances.

## 52 what is ENI (VPC)

Elastic Network Interfaces (ENIs)

An Elastic Network Interface (ENI) is a virtual network interface that you can attach to an instance in an Amazon VPC. ENIs are only available within an Amazon VPC, and they are associated with a subnet upon creation. They can have one public IP address and multiple private IP addresses. If there are multiple private IP addresses, one of them is primary. Assigning a second network interface to an instance via an ENI allows it to be dual-homed (have network presence in different subnets).

## 53 Network File System (NFS)

Link : <https://cmccloud.vn/tin-tuc/nfs-la-gi-197>

The Network File System (NFS) is a mechanism for storing files on a network. It is a distributed file system that allows users to access files and directories located on remote computers and treat those files and directories as if they were local.

## 54 What is Amazon Athena?

<https://docs.aws.amazon.com/athena/latest/ug/what-is.html>

is ServerLess (no Server => no config and managed)

Amazon Athena is an interactive query service that makes it easy to analyze data directly in Amazon Simple Storage Service (Amazon S3) using standard [SQL](https://docs.aws.amazon.com/athena/latest/ug/ddl-sql-reference.html). With a few actions in the AWS Management Console, you can point Athena at your data stored in Amazon S3 and begin using standard SQL to run ad-hoc queries and get results in seconds.

## 55 what is AWS Direct Connect

<https://aws.amazon.com/vi/directconnect/>

Direct Connect connections are provisioned on standalone hardware devices that enables you to create highly resilient network connections between Amazon Virtual Private Cloud and your on-premises infrastructure. This capability enables you to access your AWS resources in a reliable, scalable, and cost-effective way.

Dịch vụ đám mây AWS Direct Connect là đường dẫn ngắn nhất đến các tài nguyên AWS của bạn. Trong khi chuyển tiếp, lưu lượng mạng của bạn vẫn ở trên mạng toàn cầu AWS và không dùng đến Internet công cộng. Điều này giúp giảm khả năng gặp phải sự cố tắc nghẽn hoặc tăng độ trễ bất ngờ. Khi tạo kết nối mới, bạn có thể chọn một kết nối lưu trữ do Đối tác phân phối AWS Direct Connect cung cấp hoặc chọn một kết nối chuyên dụng từ AWS và triển khai tại các vị trí AWS Direct Connect trên toàn thế giới. Với AWS Direct Connect SiteLink, bạn có thể gửi dữ liệu giữa các vị trí AWS Direct Connect để tạo kết nối mạng riêng giữa các văn phòng và trung tâm dữ liệu trong mạng toàn cầu của mình.

## 56 AWS Config

AWS Config is a fully managed service that provides you with an AWS resource inventory, configuration history, and configuration change notifications to enable security and governance. With AWS Config, you can discover existing and deleted AWS resources, determine your overall compliance against rules, and dive into configuration details of a resource at any point in time. These capabilities enable compliance auditing, security analysis, resource change tracking, and troubleshooting.

<https://aws.amazon.com/vi/config/>

* Liên tục đánh giá, giám sát và ghi lại các thay đổi cấu hình tài nguyên để đơn giản hóa hoạt động quản lý thay đổi.
* Liên tục kiểm tra và đánh giá sự tuân thủ với các chính sách do tổ chức ban hành của cấu hình tài nguyên của bạn.
* Đơn giản hóa hoạt động khắc phục sự cố vận hành bằng cách tạo mối tương quan giữa các thay đổi cấu hình với các sự kiện cụ thể trong tài khoản của bạn.

### Hợp lý hóa việc khắc phục sự cố hoạt động và quản lý thay đổi

Khám phá các tài nguyên tồn tại trong tài khoản của bạn hoặc phát hành dữ liệu cấu hình của các tài nguyên bên thứ ba vào AWS Config, ghi lại cấu hình của chúng và nắm bắt mọi thay đổi để nhanh chóng khắc phục các sự cố hoạt động.

## 57 Amazon Inspector

<https://aws.amazon.com/vi/inspector/pricing/#:~:text=Amazon%20Inspector%20l%C3%A0%20m%E1%BB%99t%20d%E1%BB%8Bch,tu%C3%A2n%20th%E1%BB%A7%20cho%20kh%E1%BB%91i%20l%C6%B0%E1%BB%A3ng>

là một dịch vụ quét lỗ hổng bảo mật tự động và liên tục nhằm đánh giá các phiên bản Đám mây điện toán linh hoạt (EC2) của Amazon, hàm AWS Lambda và hình ảnh bộ chứa trong Amazon ECR và trong các công cụ tích hợp liên tục và phân phối liên tục (CI/CD) để cải thiện tính bảo mật và tuân thủ cho khối lượng ...

## 58 what is AWS Secrets Manager

<https://aws.amazon.com/vi/secrets-manager/faqs/#:~:text=AWS%20Secrets%20Manager%20enables%20you,unauthorized%20users%20viewing%20sensitive%20information>.

AWS Secrets Manager enables you to store, retrieve, control access to, rotate, audit, and monitor secrets centrally. You can encrypt secrets at rest to reduce the likelihood of unauthorized users viewing sensitive information.

* Must encrypt

 **Automatic Secret Rotation**: Allows you to automatically rotate secrets without disrupting your applications.

 **Cross-Region Replication**: Enables replication of secrets across multiple AWS regions to support disaster recovery and multi-region applications.

 **Higher Storage Limits**: Supports secrets up to 64KB in size.

## 59 what is AWS Systems Manager Parameter Store

<https://docs.aws.amazon.com/managedservices/latest/userguide/sys-man-param-store.html#:~:text=AWS%20Systems%20Manager%20Parameter%20Store%20provides%20secure%2C%20hierarchical%20storage%20for,license%20codes%20as%20parameter%20values>.

AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, and license codes as parameter values.

Not have rote key param, encrypt is optional

 **Standard Parameters**: Allow storage of values up to 4KB in size. The first 10,000 standard parameters are free.

 **Advanced Parameters**: Support values up to 8KB and offer additional features like parameter policies, but they incur additional costs.

## 60 ****What is AWS CloudFormation?****

AWS CloudFormation enables you to automate the provisioning of AWS infrastructure using code. Instead of manually creating resources (like EC2, S3, or RDS), you define them in a CloudFormation **template**, and AWS handles the provisioning.

### **Key Concepts**

* **Template**: A JSON or YAML file that describes the AWS resources.
* **Stack**: A collection of resources created from a template.
* **StackSet**: Used to deploy stacks across multiple AWS accounts and regions.
* **Change Set**: Shows the differences before updating a stack.
* Stack updates: First the changes are submitted and compared with the current state of the stack and only the changed resources get updated. There are two methods for updating stacks:

● Direct update - when there is a need to quickly deploy the updates.

● Creating and executing change sets - they are JSON files, providing a preview option for the changes to be applied. StackSets are responsible for safely provisioning, updating, or deleting stacks.

- **Nested Stacks** are stacks created within another stack by using the AWS::CloudFormation::Stack resource

**- AWS CloudFormation Registry** helps to provision third-party application resources alongside AWS resources. Examples of third-party resources are incident management, version control tools.

## 61 AWS Organizations What are AWS Organizations?\

AWS Organizations is a global service that enables users to consolidate and manage multiple AWS accounts into an organization. It includes account management and combined billing capabilities that help to meet the budgetary, and security needs of the business better.

● The main account is the management account – it cannot be changed.

● Other accounts are member accounts that can only be part of a single organization

## 62 AWS Systems Manager What is AWS Systems manager?

AWS Systems Manager is a service which helps users to manage EC2 and on-premises systems at scale. It not only detects the insights about the state of the infrastructure but also easily detects problems as well. Additionally, we can patch automation for enhanced compliance. This AWS service works for both Windows and Linux operating systems

Features:

● Easily integrated with CloudWatch metrics/dashboards and AWS Config.

● It helps to discover and audit the software installed.

● Compliance management

● We can group more than 100 resource types into applications, business units, and environments.

● It helps to view instance information such as operating system patch levels, install software and see the compliance with the desired state.

● Act associate and configurations with resources and find out the discrepancies.

● Distribute multiple software versions safely across the instances.

● Increase the security area by running a command or maintaining scripts.

● Patch your instances of schedule to keep them compliant.

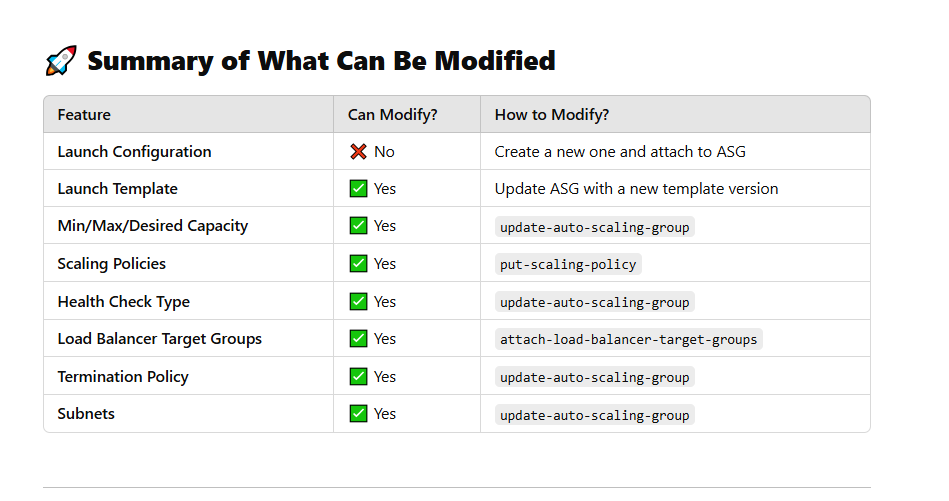
● Helps managers to automate workflows.

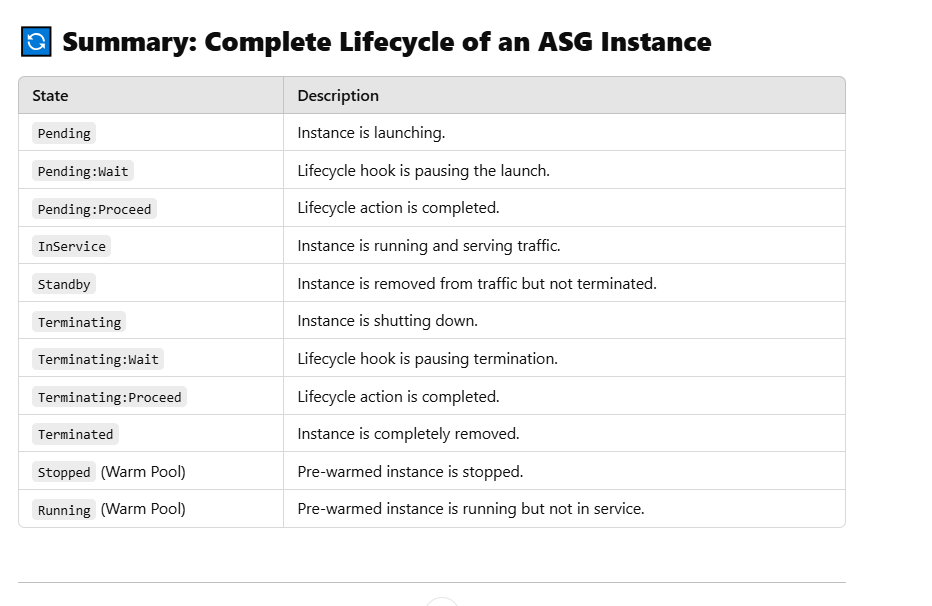
● It helps to reduce errors by securely applying configurable parameters into centralized service

**How does the System Manager work?**

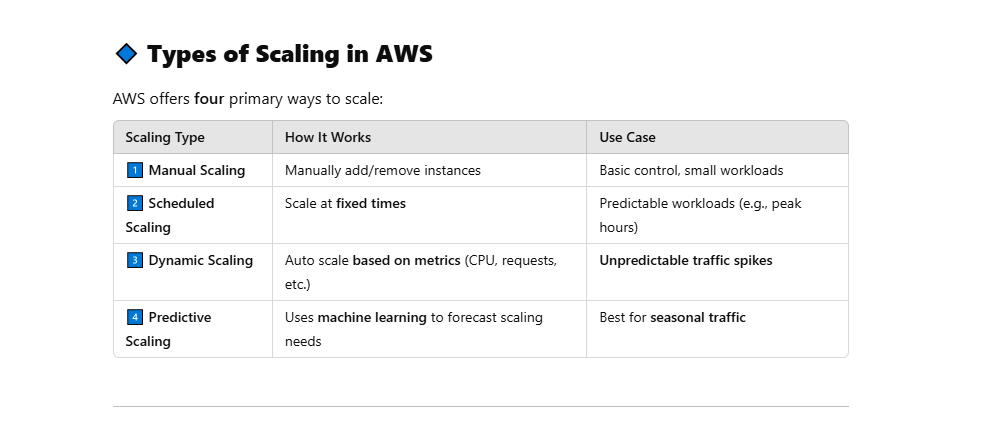
Firstly, User needs to install the SSM agent on the system they control. If an instance can’t be controlled with SSM, it’s probably an issue with the SSM agent. Also, we ne

## \*63 Auto scaling group

* It is not possible to modify a launch configuration once it is created
* 



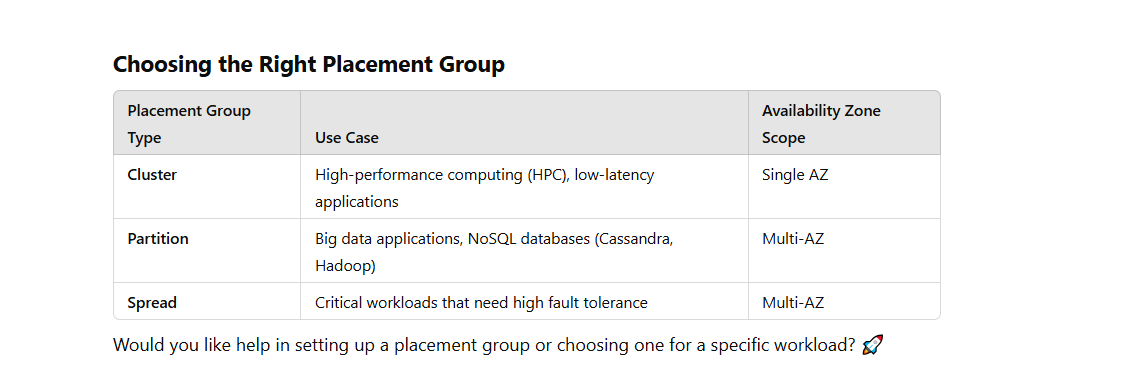
## 64 Types of Scaling in AWS



## 65 Placement Groups

### **Types of Placement Groups**

1. **Cluster Placement Group**
   * Packs instances close together inside a **single Availability Zone**.
   * Used for **low-latency, high-bandwidth communication** between instances.
   * Suitable for **HPC (High-Performance Computing)** workloads and applications requiring fast network speeds (10 Gbps to 100 Gbps networking).
   * Risk: If the hardware fails, all instances might be impacted.
2. **Partition Placement Group**
   * Spreads instances across **logical partitions** to ensure that groups of instances do not share the same underlying hardware.
   * Used for **distributed and big data applications** (e.g., Hadoop, HBase, Cassandra).
   * Reduces the impact of hardware failure by isolating partitions.
   * Works across **multiple Availability Zones** in the same Region.
3. **Spread Placement Group**
   * Ensures that **each instance is placed on distinct hardware** to improve fault tolerance.
   * Used for **small critical workloads** that must be isolated (e.g., highly available applications).
   * Can span **multiple Availability Zones**, unlike Cluster Placement Groups.
   * Limited to **7 running instances per Availability Zone**.



## 66 Amazon FSx: Types & Features

### **1. Introduction to Amazon FSx**

Amazon FSx is a managed file system service that provides fully managed file storage optimized for specific workloads. It eliminates the need to manage file servers and storage infrastructure, allowing you to set up, run, and scale file storage in AWS.

AWS offers different FSx options optimized for different workloads.

### **2. Types of Amazon FSx**

AWS provides four types of FSx, each catering to specific use cases:

#### **a) Amazon FSx for Windows File Server**

* A fully managed Windows-based file system.
* Supports **SMB protocol** for file sharing across Windows, Linux, and macOS.
* Integrated with **Active Directory (AD)** for authentication.
* Supports **shadow copies** (previous versions), encryption, and data deduplication.

**Best for:** Enterprise applications that require Windows file storage, including Microsoft SQL Server, SharePoint, and home directories.

#### **b) Amazon FSx for Lustre**

* A high-performance file system designed for compute-intensive workloads.
* Provides ultra-low latency and high throughput.
* Supports **integration with Amazon S3**, allowing automatic data synchronization.
* Used for **machine learning (ML), big data analytics, and high-performance computing (HPC).**

**Best for:** Workloads requiring fast, parallelized processing such as financial modeling, genomic analysis, and AI training.

#### **c) Amazon FSx for NetApp ONTAP**

* A managed NetApp ONTAP file system in AWS.
* Supports **NFS, SMB, and iSCSI protocols**.
* Provides **snapshots, data deduplication, and replication**.
* Compatible with on-premises NetApp ONTAP environments for hybrid cloud storage.

**Best for:** Enterprises using NetApp storage solutions and requiring hybrid cloud deployment.

#### **d) Amazon FSx for OpenZFS**

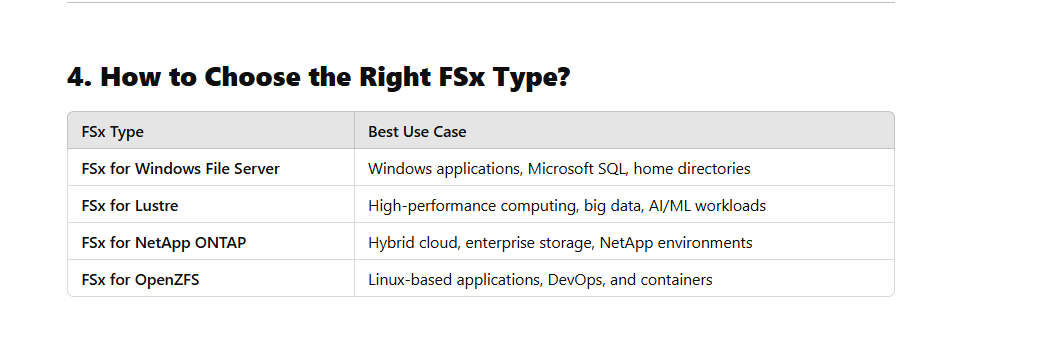
* A fully managed file system based on **OpenZFS**, a high-performance storage solution.
* Provides **automatic compression, snapshots, and high-speed storage**.
* Supports **NFS protocol** for Linux-based applications.

**Best for:** Linux workloads, including DevOps, containerized applications, and database workloads.

### **3. Key Features of Amazon FSx**

Each Amazon FSx type has unique features, but the common benefits include:

✅ **Fully Managed:** AWS handles provisioning, patching, and scaling.  
✅ **High Performance:** Optimized for low-latency and high-throughput applications.  
✅ **Multi-Protocol Support:** Supports SMB, NFS, Lustre, and iSCSI for flexible integration.  
✅ **Security & Compliance:** Supports encryption, VPC integration, and compliance standards like HIPAA and GDPR.  
✅ **Hybrid Cloud Support:** Integrates with on-premises environments using AWS Direct Connect or VPN.  
✅ **Backup & Restore:** Supports automatic backups and snapshots for data protection.



## **66.1 FSx for Lustre (FSx for LSX)** storage types

## ****1. Persistent Storage (Long-Term Use)****

🔹 Designed for **high durability and long-term storage**.  
🔹 Data is replicated within the AWS Availability Zone.  
🔹 Best for applications requiring **consistent availability and reliability**.

### **Persistent Storage Classes**

| **Storage Class** | **Durability** | **Performance** | **Pricing** |
| --- | --- | --- | --- |
| **Persistent SSD (PSSD)** | Multi-AZ durability | **High IOPS & low latency** | ~$0.14 per GB-month |
| **Persistent HDD (PHDD)** | Multi-AZ durability | **Lower performance, cost-effective** | ~$0.025 per GB-month |

✅ **Use Case:** Big data analytics, AI/ML workloads, long-running HPC simulations.

## ****2. Scratch Storage (Temporary Use)****

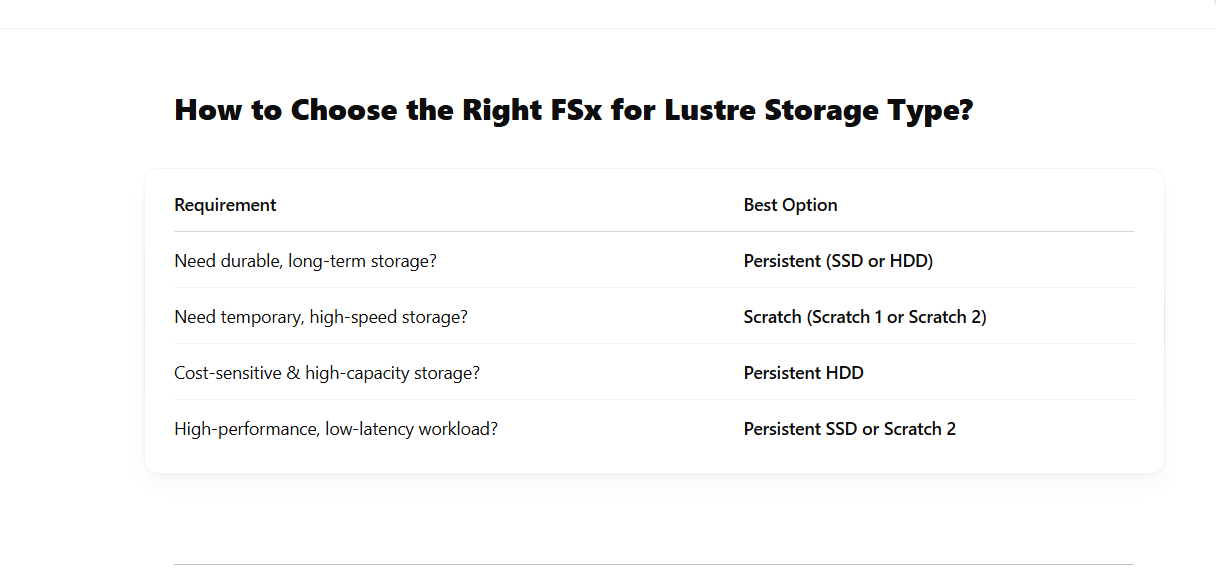
🔹 Designed for **temporary, high-speed processing**.  
🔹 Data is **NOT** replicated—if a failure occurs, data is lost.  
🔹 Best for workloads that can regenerate data if needed.

### **Scratch Storage Classes**

| **Storage Class** | **Durability** | **Performance** | **Pricing** |
| --- | --- | --- | --- |
| **Scratch 1** | No redundancy | **High throughput, parallel access** | ~$0.035 per GB-month |
| **Scratch 2** | No redundancy | **Better performance, burst mode support** | ~$0.14 per GB-month |

✅ **Use Case:** Short-term processing of machine learning datasets, video rendering, and scientific simulations.

## ****How to Choose the Right FSx for Lustre Storage Type?****



## 67 what is power the live leaderboard. ?

A **live leaderboard** is powered by a **high-speed database system** that can handle **real-time updates, sorting, and ranking** efficiently. The key components that power a live leaderboard are:

### 🔥 **Key Technologies for Powering a Live Leaderboard**

1️⃣ **In-Memory Databases (For Speed & Ranking)**

* **ElastiCache (Redis) – Best for Real-Time Leaderboards**
  + Uses **Sorted Sets (ZSETs)** for ranking players dynamically.
  + **Super-fast lookups** (sub-millisecond latency).
  + Can efficiently update scores and retrieve **top N players**.
  + Example:

sh

ZADD leaderboard 1001 5000 # Add user 1001 with score 5000

ZREVRANGE leaderboard 0 9 WITHSCORES # Get Top 10 players

2️⃣**NoSQL Databases (For Scalability & Persistence)**

* **Amazon DynamoDB** (for long-term storage & large-scale leaderboards)
  + Stores user scores with a **highly scalable key-value structure**.
  + Uses **Global Secondary Indexes (GSI)** to query leaderboards efficiently.
  + Can work with **DynamoDB Streams + Lambda** for real-time updates.
  + Example Schema:

json

{ "userId": "1001", "score": 5000, "timestamp": "2025-03-05T10:00:00Z" }

3**Streaming & Event Processing (For Live Updates)**

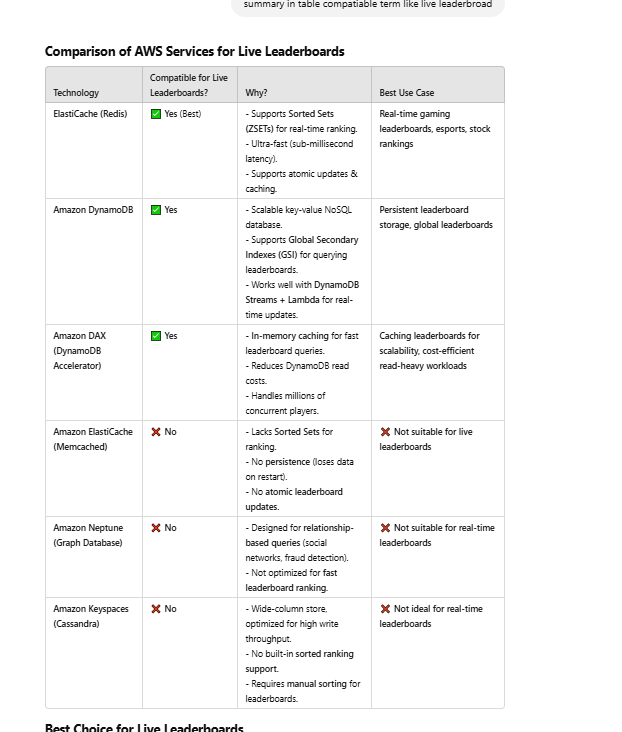
* **AWS Lambda + DynamoDB Streams** (for real-time event-driven updates).
* **AWS Kinesis / Kafka** (for high-volume leaderboard updates).
* **WebSockets / Socket.IO** (to push leaderboard updates to clients instantly).

4️**Hybrid Approach (Best of Both Worlds)**

* Store leaderboard data **persistently** in **DynamoDB**.
* Use **ElastiCache (Redis)** as a **cache layer** for ultra-fast lookups.
* Use **WebSockets** or **API Gateway** to send updates to frontend apps.



Summary



## 68 why elastic memcache is not live leaderbread

### ❌ Why **ElastiCache Memcached** is NOT Ideal for a Live Leaderboard?

ElastiCache supports two in-memory databases: **Redis** and **Memcached**.  
However, **Memcached is NOT suitable for a live leaderboard** because:

### 🔴 **1. No Built-in Sorted Sets (ZSETs)**

* Redis provides **Sorted Sets (ZSETs)**, which automatically maintain ranking.
* Memcached does **not** have an equivalent feature, so you need to **manually sort data**, which is slow.

### 🔴 **2. No Persistence (Volatile Data)**

* Memcached **does not persist data to disk**.
* If the instance restarts, **all leaderboard data is lost**.
* Redis, on the other hand, supports **RDB (snapshotting) and AOF (Append-Only File)** to recover data.

### 🔴 **3. No Atomic Operations for Leaderboards**

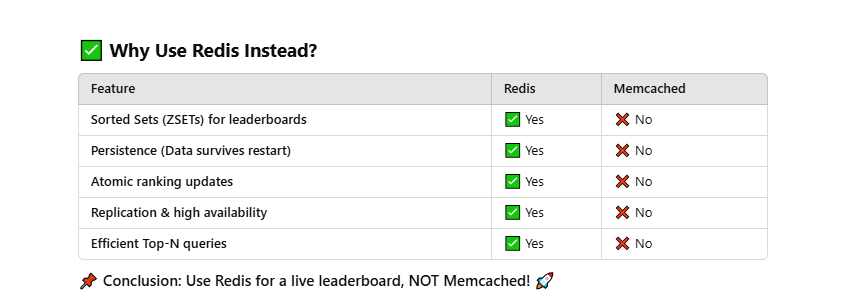
* In Redis, operations like ZADD, ZREVRANGE, and ZINCRBY are **atomic**, meaning no two updates can interfere.
* Memcached lacks **transaction-like** atomic operations, making updates unreliable in real-time scenarios.

### 🔴 **4. No Built-in Replication & High Availability**

* **Redis supports replication (Master-Slave) and failover**, ensuring high availability.
* Memcached lacks **native replication**, so if a node fails, **data is lost**.

### 🔴 **5. No Efficient Range Queries**

* In Redis, you can efficiently get **Top 10 players** with ZREVRANGE leaderboard 0 9.
* Memcached requires **fetching all keys and sorting them manually**, which is **slow and inefficient**.



## 69 VPC Endpoint

### **1. VPC Endpoint**

A **VPC Endpoint** allows you to connect to AWS services **privately** within your VPC. It eliminates the need for an Internet Gateway, NAT Gateway, or VPN.

There are two types of VPC Endpoints:

* **Interface Endpoint**
* **Gateway Endpoint**

### **2. Interface Endpoint**

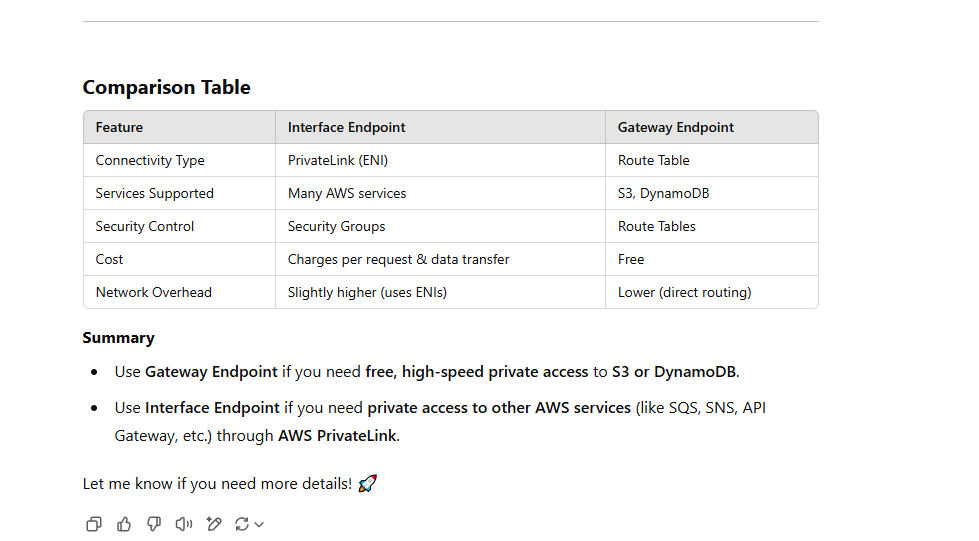
* Uses **AWS PrivateLink** to provide private connectivity.
* Deploys **Elastic Network Interfaces (ENIs)** inside your VPC.
* Supports AWS services like **S3, DynamoDB, SNS, SQS, and many more**.
* Requires **Security Group rules** to control access.
* Support only a AZ

✅ **Use Case**: You need private access to AWS services that are normally accessed via public endpoints, such as **AWS S3, DynamoDB, or AWS Lambda**.

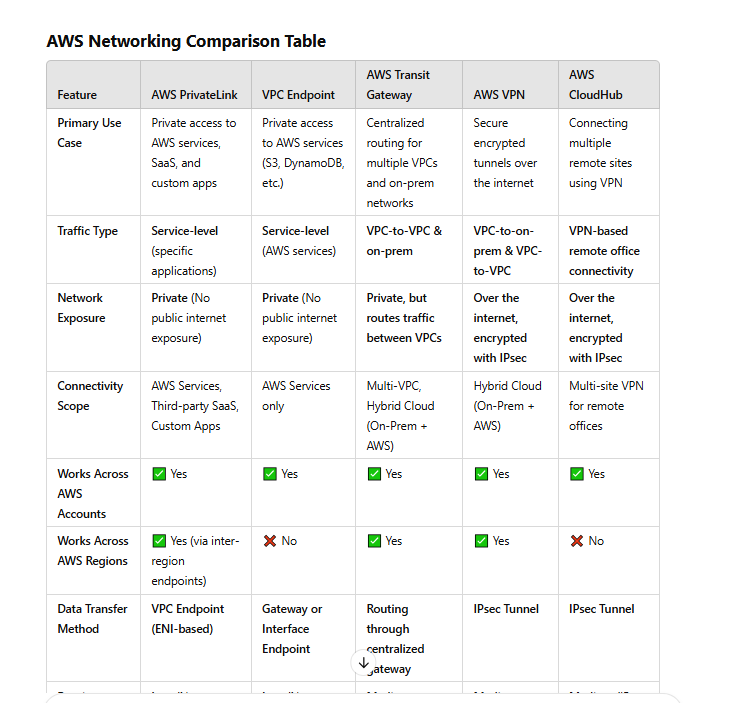
### **3. Gateway Endpoint**

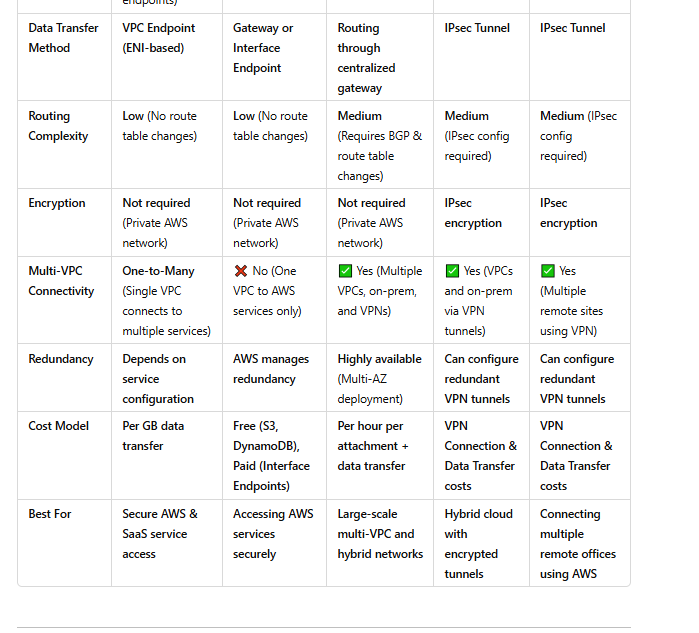
* Specifically designed for **Amazon S3 and DynamoDB**.
* Uses **route tables** instead of ENIs. => phụ thuộc vào VPC => có thể multi AZ
* Traffic is directed through the **VPC route table** instead of the internet.

✅ **Use Case**: You need private access to **S3 or DynamoDB** without using NAT or an internet gateway.



# connection optional

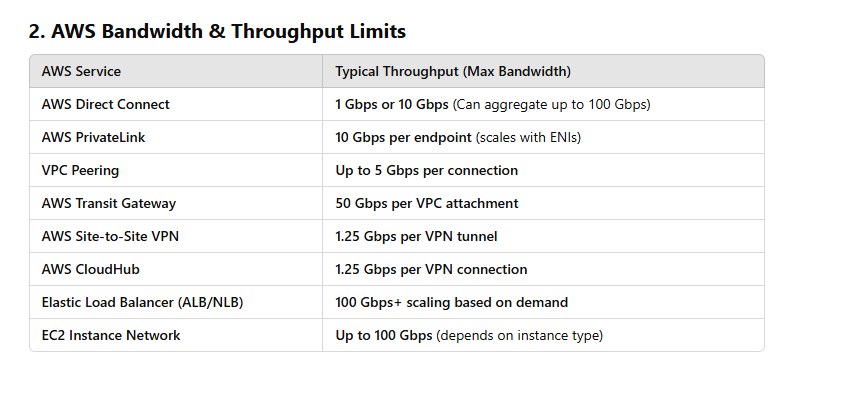




### Cost compare



### Thoughtput and bandwidth



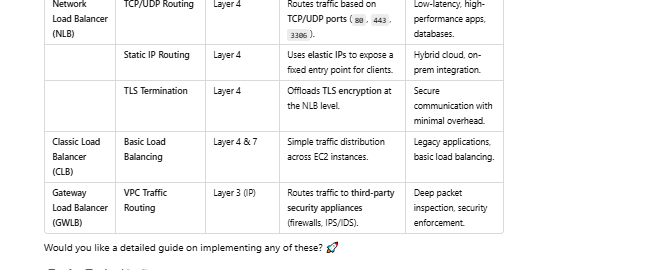
### **3. Key Considerations for Throughput in AWS**

✅ **PrivateLink vs. VPN:** PrivateLink offers **higher bandwidth** and **lower latency** compared to VPN.  
✅ **Direct Connect:** Best for **high-throughput** applications (AWS to on-prem).  
✅ **Transit Gateway:** Best for **multi-VPC** communication with **higher throughput than VPC Peering**.  
✅ **Scaling:** For higher bandwidth, **aggregate multiple connections** (e.g., multiple PrivateLink endpoints, Direct Connect links).

Would you like recommendations for optimizing network throughput? 😊

# Detailed Routing of Application Load Balancer (ALB) in AWS

### 



### **Overview of ALB Routing**

An **Application Load Balancer (ALB)** in AWS operates at **Layer 7 (HTTP/HTTPS)** and intelligently routes traffic based on URLs, host headers, query strings, request methods, and more. It is commonly used to distribute traffic among instances in **Auto Scaling Groups (ASGs)**.

ALB consists of:

1. **Listeners** – Rules that define how requests are handled (e.g., HTTP/HTTPS).
2. **Rules** – Conditions that determine where the request should be routed.
3. **Target Groups** – The backend destinations (EC2, Lambda, ECS, or IPs).
4. **Health Checks** – Ensure only healthy targets receive traffic.

## ****1. ALB Routing Types****

### **A. Path-Based Routing**

* Routes traffic based on the **request path** (URL).
* Useful for **microservices** and **monolithic applications** with multiple modules.

#### **Example**

If you have two services:

* example.com/api/\* → Routes to an ASG with backend API servers.
* example.com/app/\* → Routes to an ASG with frontend web servers.

##### **ALB Rule Configuration**

| **Condition** | **Action** |
| --- | --- |
| Path is /api/\* | Forward to api-target-group |
| Path is /app/\* | Forward to frontend-target-group |

### **B. Host-Based Routing**

* Routes traffic based on the **Host Header** (domain name).
* Useful for **multi-tenant applications** or serving different applications under the same ALB.

#### **Example**

* api.example.com → Routes to API servers.
* admin.example.com → Routes to an admin dashboard.

##### **ALB Rule Configuration**

| **Condition** | **Action** |
| --- | --- |
| Host is api.example.com | Forward to api-target-group |
| Host is admin.example.com | Forward to admin-target-group |

### **C. Query String & Header-Based Routing**

* Routes traffic based on **query parameters** or **headers** in the request.
* Useful for **A/B testing** or feature toggles.

#### **Example**

* example.com/?version=v2 → Routes to a new ASG.
* Requests with Header: User-Agent = Mobile → Routes to mobile-optimized servers.

##### **ALB Rule Configuration**

| **Condition** | **Action** |
| --- | --- |
| Query String version=v2 | Forward to v2-target-group |
| Header User-Agent=Mobile | Forward to mobile-target-group |

### **D. HTTP to HTTPS Redirection**

* Redirects HTTP traffic to HTTPS for better security.

##### **ALB Rule Configuration**

| **Condition** | **Action** |
| --- | --- |
| HTTP Request | Redirect to HTTPS (301) |

### **E. Fixed Response Routing**

* Returns a static response without forwarding to any backend.
* Useful for **maintenance mode pages** or blocking unwanted requests.

##### **ALB Rule Configuration**

| **Condition** | **Action** |
| --- | --- |
| Path is /maintenance | Return 503 Service Unavailable |

## ****2. Target Groups & Health Checks****

Each ALB rule **forwards traffic to a Target Group**, which can contain:

* **EC2 instances (Auto Scaling Group)**
* **ECS (Containers)**
* **Lambda Functions**
* **IP Addresses (On-premises or hybrid setups)**

### **Health Checks**

ALB performs **health checks** to ensure only healthy instances receive traffic.

| **Parameter** | **Description** |
| --- | --- |
| Protocol | HTTP or HTTPS |
| Path | /health (default) |
| Interval | 30s (adjustable) |
| Threshold | Fails if consecutive health check failures exceed limit |

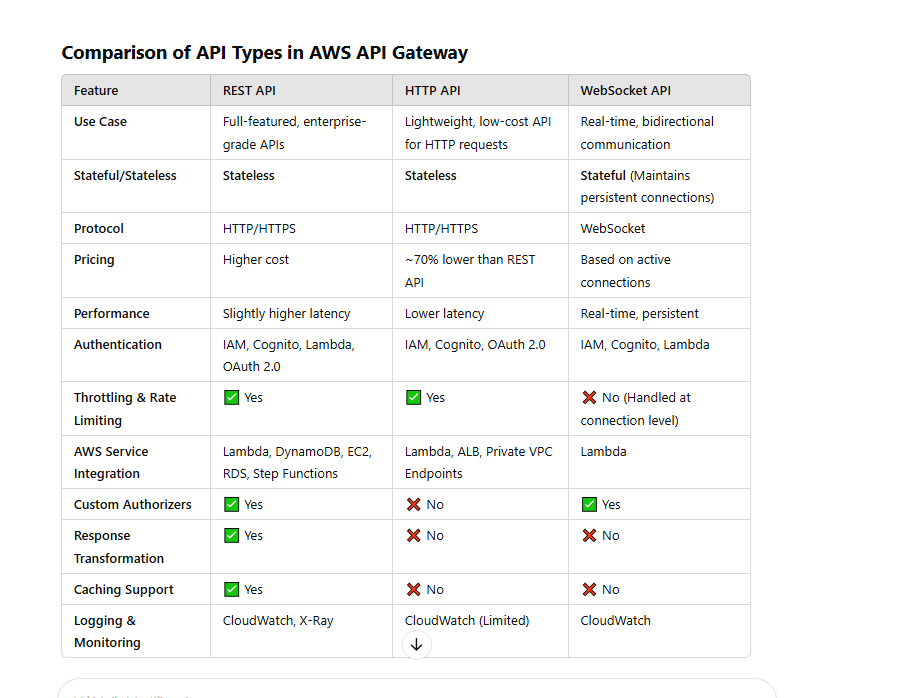
## ****3. ALB Sticky Sessions (Session Affinity)****

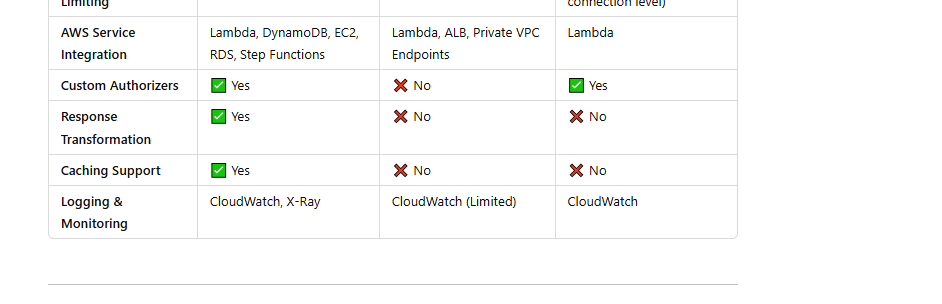
* Ensures a user’s requests go to the same instance for session consistency.
* Uses **Application Cookie (AWSALB)** or **Duration-Based Stickiness**.

## ****4. ALB Authentication & Security Features****

* **OIDC Authentication** (e.g., Google, Okta, Cognito)
* **WAF (Web Application Firewall)** for security filtering
* **Access Logs & CloudWatch Monitoring**

# AWS API Gateway





# ****73 Additional Failover Control****

Today we are making Aurora even more flexible by giving you control over the failover priority of each read replica. Each read replica is now associated with a priority tier (0-15).  In the event of a failover, Amazon RDS will promote the read replica that has the highest priority (the lowest numbered tier).

# promote in aurora

* Amazon Aurora features a distributed, fault-tolerant, self-healing storage system that auto-scales up to 64TB per database instance. It delivers high performance and availability with up to 15 low-latency read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across three Availability Zones (AZs).
* For Amazon Aurora, each read replica is associated with a priority tier (0-15). In the event of a failover, Amazon RDS will promote the read replica that has the highest priority (the lowest numbered tier). If two or more Aurora Replicas share the same priority, then Amazon RDS promotes the replica that is largest in size. If two or more Aurora Replicas share the same priority and size, then Amazon RDS promotes an arbitrary replica in the same promotion tier. Therefore, for this problem statement, the Tier-1 (32TB) replica will be promoted. Reference: https://aws.amazon.com/blogs/aws/additional-failover-control-for-amazon-aurora/ <https://docs.amazonaws.cn/en_us/AmazonRDS/latest/AuroraUserGuide/Aurora.Managing.Backups.html#Aurora.Managing.FaultTolerance>

# 75 what is caching layer in aws

A **caching layer** in AWS is a mechanism used to temporarily store frequently accessed data in memory to improve performance and reduce latency. This helps applications avoid repeated database queries or API calls, leading to faster response times and reduced infrastructure costs.

### **Common Caching Services in AWS**

AWS provides multiple caching options, depending on the use case:

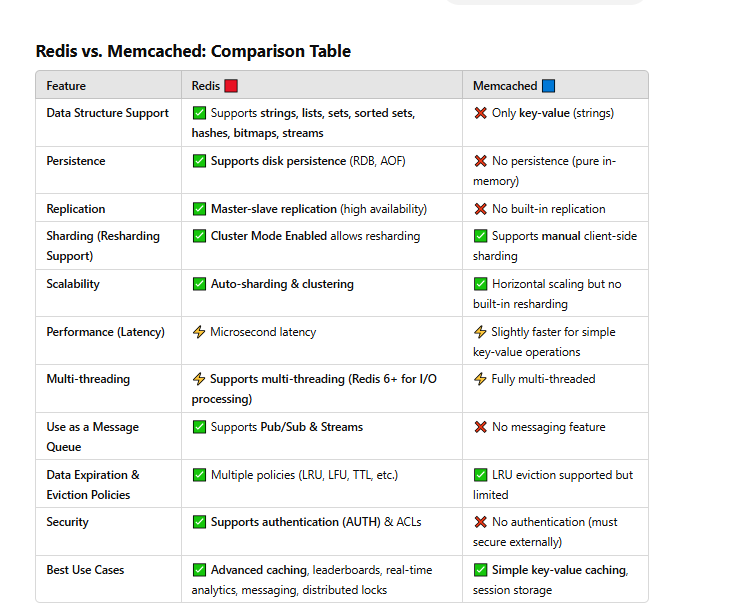
1. **Amazon ElastiCache** (Redis & Memcached)
   * In-memory data store for high-speed caching.
   * Supports two caching engines:
     + **Redis**: Advanced key-value store with persistence, pub/sub, and replication.
     + **Memcached**: Simple, high-performance caching layer.
2. **AWS CloudFront** (Content Delivery Network)
   * Caches static and dynamic content at edge locations worldwide.
   * Reduces latency by serving content closer to users.
3. **AWS Lambda Cache**
   * Stores data in-memory within the execution environment to reduce cold start times.
4. **Amazon RDS Proxy**
   * Caches database connections to improve performance for high-concurrency workloads.
5. **Amazon DynamoDB Accelerator (DAX)**
   * A caching layer for DynamoDB that reduces response times from milliseconds to microseconds.

### **When to Use a Caching Layer**

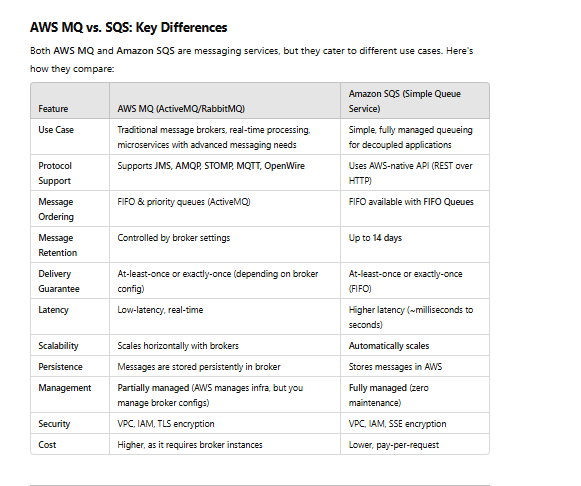
* Reduce database or API load.
* Improve application performance by minimizing data retrieval time.
* Enhance scalability for high-traffic applications.

Would you like help choosing the best caching strategy for your use case?

# 76 key cache for reshading (Redis vs. Memcached: Comparison Table)



# 77 AWS MQ Overview



### **When to Use AWS MQ?**

✅ **If you need JMS, AMQP, MQTT, or STOMP support** (e.g., migrating from on-prem ActiveMQ).  
✅ **If your application requires priority queues, message ordering, or pub-sub models** beyond SNS/SQS.  
✅ **If you need long-lived connections** (e.g., chat applications, real-time event processing).

### **When to Use SQS?**

✅ **If you want a simple, fully managed queue** that scales automatically.  
✅ **If your app doesn’t rely on complex broker features like message prioritization.**  
✅ **If cost is a concern** and you prefer a pay-per-message model.

Would you like help deciding which one fits your specific use case? 🚀

# 78 What is Elastic Fabric Adapter

Elastic Fabric Adapter (EFA) is a **high-performance network interface** designed by AWS to accelerate **High-Performance Computing (HPC) and Machine Learning (ML) workloads** on EC2 instances. It provides **low-latency, high-bandwidth communication** between instances, making it ideal for workloads that require tightly coupled, distributed computing.

### Key Features of EFA:

1. **Low Latency & High Throughput**
   * Uses **AWS Scalable Reliable Datagram (SRD)** protocol for fast inter-instance communication.
   * Reduces network overhead compared to traditional TCP-based networking.
2. **Supports MPI & NCCL Applications**
   * Compatible with **Message Passing Interface (MPI)** for parallel computing.
   * Works with **NVIDIA Collective Communications Library (NCCL)** for ML training.
3. **Elastic Network Adapter (ENA) Compatibility**
   * Extends standard **ENA** but adds hardware-based acceleration for HPC workloads.
4. **Designed for HPC and ML Training**
   * Suitable for applications like computational fluid dynamics (CFD), seismic analysis, financial modeling, and deep learning.
5. **Scales Across Multiple Instances**
   * Works in **placement groups** to ensure low-latency communication between instances.

### How to Use EFA:

1. **Launch an EC2 instance** with EFA-enabled instance types (e.g., C5n, P3dn, P4, Hpc6a).
2. **Enable EFA in the instance’s network interface** via AWS Management Console or CLI.
3. **Install AWS EFA software** (efa-installer) on the instance.
4. **Use MPI or NCCL libraries** for high-performance networking.

### When to Use EFA:

✅ HPC workloads requiring low-latency interconnect  
✅ Machine learning distributed training  
✅ Computational simulations needing fast inter-node communication  
✅ Applications where traditional TCP networking is a bottleneck

Would you like details on configuring EFA in your EC2 instances?

# 79 Third-Party Configuration Management Tools in AWS

AWS also supports popular **third-party configuration management tools** that integrate with AWS services.

## ****1. Chef****

Chef is a **procedural configuration management tool** that automates infrastructure and application deployment. It follows a **"cookbook and recipe"** approach to define configurations.

### **How Chef Works in AWS**

* Uses **Chef Workstation** to create configuration scripts.
* Chef configurations are written in **Ruby-based DSL**.
* AWS instances (nodes) use a **Chef Client** to fetch and apply configurations from a **Chef Server**.
* AWS provides **OpsWorks for Chef Automate**, a managed Chef server.

### **Key Features**

* **Cookbooks & Recipes**: Define infrastructure and application setup using Ruby scripts.
* **Chef Client & Chef Server**: Nodes pull configurations from the central Chef Server.
* **Knife CLI**: Tool to manage configurations from the command line.
* **Chef Automate**: Provides compliance auditing, analytics, and reporting.

### **Use Cases**

* Automating application deployments on AWS EC2 instances.
* Managing configurations across AWS, on-premises, and multi-cloud environments.
* Ensuring infrastructure compliance using **Chef InSpec**.

### **AWS Integration**

* **AWS OpsWorks for Chef Automate** (Managed Chef Server).
* **AWS SDK for Chef** to automate EC2, RDS, and other AWS resources.
* **Amazon EC2 Systems Manager** integration for automating patching.

## ****2. Puppet****

Puppet is a **declarative configuration management tool** that ensures system states are maintained. It uses a **"desired state"** model, meaning administrators define what the system should look like, and Puppet enforces that state.

### **How Puppet Works in AWS**

* Uses a **Puppet Master** server to store configurations.
* AWS instances run a **Puppet Agent** that checks in with the master and applies necessary changes.
* AWS provides **OpsWorks for Puppet Enterprise**, a managed Puppet service.

### **Key Features**

* **Declarative Language**: Defines what the infrastructure should look like, and Puppet ensures compliance.
* **Puppet Modules**: Predefined configuration modules to set up common services.
* **Node Classification**: Applies different configurations to different instance groups.
* **Puppet Forge**: A repository of reusable modules for AWS and other platforms.

### **Use Cases**

* Enforcing security policies on AWS instances (e.g., firewall rules, SSH settings).
* Managing **multi-cloud environments** (AWS, Azure, Google Cloud).
* Automating OS patching and application updates.

### **AWS Integration**

* **AWS OpsWorks for Puppet Enterprise** (Managed Puppet Master).
* **Puppet Modules for AWS** (Prebuilt configurations for EC2, S3, IAM, etc.).
* **Amazon CloudWatch** integration for monitoring compliance.

## ****3. Ansible****

Ansible is an **agentless configuration management tool** that automates system setup, application deployment, and orchestration. It uses **YAML-based playbooks** to define tasks.

### **How Ansible Works in AWS**

* Unlike Chef and Puppet, Ansible **does not require an agent**.
* Uses SSH (Linux) or WinRM (Windows) to apply configurations directly to AWS instances.
* Ansible playbooks define tasks, and the **Ansible Control Node** executes them.

### **Key Features**

* **Agentless Architecture**: No need to install software on AWS instances.
* **Playbooks**: YAML-based configuration scripts that define setup steps.
* **Modules**: Reusable scripts for AWS services like EC2, RDS, and S3.
* **Idempotent Execution**: Ensures changes are only applied when needed.

### **Use Cases**

* Automating **multi-tier application deployments** on AWS.
* Configuring EC2 instances without installing an agent.
* Managing cloud infrastructure as code.

### **AWS Integration**

* **AWS Ansible Modules**: Automate AWS resources (e.g., launching EC2 instances, managing security groups).
* **AWS Systems Manager** integration for centralized automation.
* **Ansible Tower** for visualizing automation workflows.

## ****4. SaltStack****

SaltStack (Salt) is a **high-speed, scalable configuration management tool** that operates using a **master-minion** architecture. It is designed for real-time automation.

### **How SaltStack Works in AWS**

* **Salt Master**: Central server that defines configurations.
* **Salt Minions**: AWS instances that receive and apply configurations from the master.
* **Salt SSH**: Provides an agentless mode similar to Ansible.

### **Key Features**

* **Event-Driven Automation**: Reacts to real-time system changes.
* **State Files (SLS)**: YAML-based configuration files for defining infrastructure.
* **Remote Execution**: Runs commands across thousands of instances simultaneously.
* **High-Performance Scalability**: Efficient for large AWS deployments.

### **Use Cases**

* Managing large-scale AWS deployments with **real-time automation**.
* Automating server patching, security updates, and compliance.
* Handling complex configurations requiring **high-speed execution**.

### **AWS Integration**

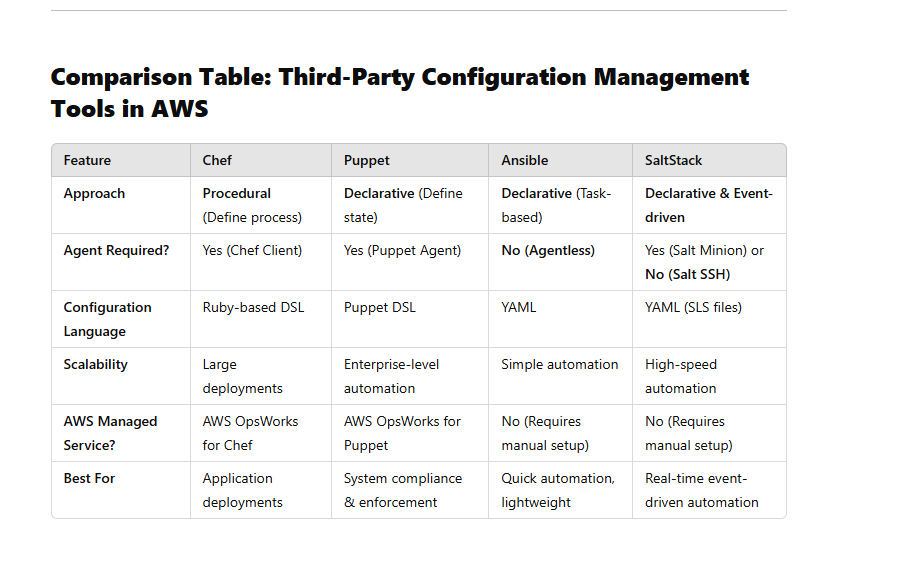
* **AWS Salt Modules**: Automate AWS services like EC2, S3, and IAM.
* **Amazon EC2 Systems Manager** integration for remote execution.
* **Salt Cloud** for managing AWS instances dynamically.

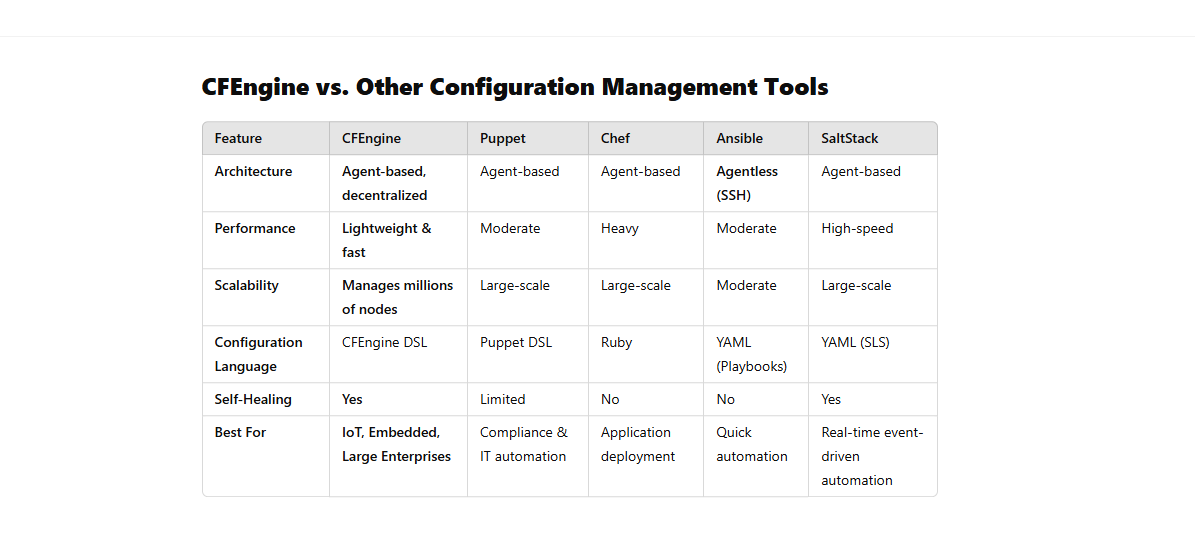
## ****CFEngine: Overview****

**CFEngine** is one of the oldest and most lightweight **configuration management tools**, designed for **automating infrastructure management** across large-scale IT environments. It follows a **declarative model**, meaning you define the desired state of your systems, and CFEngine ensures they remain in that state.

## ****Key Features of CFEngine****

1. **Lightweight and Fast**
   * Uses a **small footprint (less than 10 MB)**, making it ideal for embedded systems, IoT, and cloud environments.
   * Can handle thousands of nodes with minimal resource usage.
2. **Declarative Configuration**
   * Defines the **desired state** of servers, and CFEngine enforces compliance.
   * Uses **Promise Theory** (a mathematical model ensuring system reliability).
3. **Scalability**
   * Can manage **millions of nodes** with its distributed architecture.
   * Efficient for **large enterprises** with vast infrastructure.
4. **Security-Focused**
   * Uses **cryptographic authentication** and encrypted communication.
   * Provides **fine-grained access control**.
5. **Autonomous & Self-Healing**
   * Runs on each system as an **autonomous agent**, reducing dependency on a central server.
   * Detects **drift** (changes from the desired state) and automatically corrects it.
6. **Multi-Platform Support**
   * Works on **Linux, Unix, Windows, MacOS, and IoT devices**.





# 80 AWS **Security Hub**

is a cloud security service that helps organizations manage and improve their security posture across AWS accounts and services. It provides a **centralized view** of security findings from various AWS services like **Amazon GuardDuty, AWS Config, AWS IAM Access Analyzer, and AWS Firewall Manager**, as well as third-party security tools.

### 🔹 Key Features:

1. **Security Findings Aggregation**
   * Collects and normalizes security alerts from multiple AWS security tools and partner solutions.
2. **Compliance Standards Monitoring**
   * Automatically checks compliance with frameworks like **CIS AWS Foundations Benchmark, PCI DSS, NIST 800-53, and AWS Foundational Security Best Practices**.
3. **Automated Security Checks**
   * Uses AWS Config rules and best practices to identify misconfigurations or risks.
4. **Security Insights Dashboard**
   * Provides a **unified dashboard** to view, filter, and prioritize security issues.
5. **Integration with AWS Security Tools**
   * Works with **AWS Organizations** for multi-account management.
   * Can trigger **AWS Lambda functions** to automate responses.
6. **Custom Actions & Remediation**
   * Allows setting up **custom workflows** using AWS Security Hub rules and Amazon EventBridge.

### 🔹 How It Helps:

* Improves **visibility** into security risks across AWS workloads.
* Reduces **manual effort** in analyzing security findings.
* Enhances **incident response** by automating alerts and remediation.

# 81 Overview all security tool

## ****1️⃣ Threat Detection & Monitoring****

These tools help **identify, analyze, and respond** to security threats across AWS workloads.

### 🔹 **Amazon GuardDuty** (Intrusion Detection)

✅ Uses **machine learning & threat intelligence** to detect anomalies.  
✅ Analyzes **CloudTrail logs, VPC Flow Logs, and DNS queries** for suspicious activity.  
✅ Detects threats like **brute force attacks, data exfiltration, and compromised IAM credentials**.  
✅ **Pricing**: Based on the number of analyzed logs.

💡 **Use Case**: You can use GuardDuty to detect a hacker trying to escalate IAM permissions or accessing an S3 bucket from an unusual location.

### 🔹 **AWS Security Hub** (Centralized Security Dashboard)

✅ Aggregates security alerts from AWS services (GuardDuty, IAM Access Analyzer, Inspector).  
✅ Provides **compliance checks** for **CIS, PCI DSS, NIST, and AWS best practices**.  
✅ Supports **third-party integrations** (e.g., CrowdStrike, Palo Alto Networks).

💡 **Use Case**: If you manage multiple AWS accounts, Security Hub gives a unified **security posture overview**.

### 🔹 **Amazon Inspector** (Vulnerability Scanning)

✅ Scans **EC2 instances and containers** for known software vulnerabilities (CVEs).  
✅ Uses the **Common Vulnerability Scoring System (CVSS)** to rank risks.  
✅ Runs **automated continuous scanning** for Amazon ECS (container workloads).

💡 **Use Case**: If an EC2 instance is running outdated software with **a known security flaw**, Inspector will flag it.

### 🔹 **AWS Detective** (Threat Investigation)

✅ Uses **graph analysis & machine learning** to connect and analyze security data.  
✅ Helps **security teams investigate** suspicious activity detected by GuardDuty.  
✅ Automatically **visualizes** relationships between AWS resources, users, and threats.

💡 **Use Case**: If GuardDuty detects suspicious API calls, Detective helps trace the attack path to find the source.

## ****2️⃣ Identity & Access Management****

These tools **control user access** and **prevent unauthorized access** to AWS resources.

### 🔹 **AWS Identity and Access Management (IAM)**

✅ Manages user access to AWS services using **IAM policies, groups, and roles**.  
✅ Supports **Multi-Factor Authentication (MFA)** for extra security.  
✅ Allows **temporary security credentials** via AWS STS (Secure Token Service).

💡 **Use Case**: If you want to grant an EC2 instance access to an S3 bucket **without using hardcoded credentials**, you can use IAM roles.

### 🔹 **AWS IAM Access Analyzer**

✅ **Automatically analyzes IAM policies** to find **overly permissive access**.  
✅ Detects **publicly accessible S3 buckets, unrestricted IAM roles, and open security groups**.  
✅ Helps organizations **enforce least privilege access**.

💡 **Use Case**: If an IAM role accidentally grants **read/write access to all AWS accounts**, Access Analyzer will alert you.

### 🔹 **AWS Organizations & SCPs** (Service Control Policies)

✅ **Manages multiple AWS accounts** under a single organization.  
✅ Enforces **security policies** using SCPs (Service Control Policies).  
✅ Helps restrict access at an **organization-wide level** (e.g., block root user access).

💡 **Use Case**: If your company wants to **block all accounts from launching EC2 instances in unauthorized regions**, use SCPs.

## ****3️⃣ Network Security****

These tools protect your **network, applications, and infrastructure** from unauthorized access.

### 🔹 **AWS Shield** (DDoS Protection)

✅ Protects AWS applications from **Distributed Denial-of-Service (DDoS) attacks**.  
✅ **Shield Standard** (free) defends against common **network-layer attacks**.  
✅ **Shield Advanced** (paid) provides **real-time attack monitoring** and **DDoS cost protection**.

💡 **Use Case**: If your website is under a **DDoS attack**, Shield can block malicious traffic automatically.

### 🔹 **AWS WAF (Web Application Firewall)**

✅ Protects **web applications** from **SQL injection, XSS, and bot attacks**.  
✅ Allows defining **custom security rules** using **IP filtering, rate limiting, and pattern matching**.  
✅ Integrates with **CloudFront, API Gateway, and ALB (Application Load Balancer)**.

💡 **Use Case**: If your web app is **getting spammed with bot traffic**, AWS WAF can block **suspicious requests**.

### 🔹 **AWS Firewall Manager** (Centralized Network Security)

✅ **Centralized security policy management** for multiple AWS accounts.  
✅ Automates deployment of **AWS WAF, Shield, and Security Group policies**.  
✅ Ensures consistent **firewall rules across all AWS environments**.

💡 **Use Case**: If you manage a large AWS environment, Firewall Manager ensures **all accounts follow the same security rules**.

## ****4️⃣ Data Security & Encryption****

These tools protect **data at rest and in transit**.

### 🔹 **AWS Key Management Service (KMS)**

✅ Provides **encryption key management** for AWS services.  
✅ Uses **AWS-managed keys or customer-managed keys (CMKs)**.  
✅ Encrypts data in **S3, RDS, EBS, and Lambda**.

💡 **Use Case**: You can encrypt sensitive data stored in an **RDS database** using a **customer-managed KMS key**.

### 🔹 **AWS Secrets Manager**

✅ Securely stores and rotates **database credentials, API keys, and passwords**.  
✅ Eliminates the need for **hardcoded credentials** in applications.  
✅ Supports **automatic password rotation** for Amazon RDS.

💡 **Use Case**: Instead of **storing database passwords in code**, use **Secrets Manager** to fetch credentials securely.

### 🔹 **AWS Certificate Manager (ACM)**

✅ Issues and manages **SSL/TLS certificates** for AWS services.  
✅ Automates **certificate renewal and deployment** for secure HTTPS connections.  
✅ Works with **CloudFront, ALB, API Gateway, and Elastic Beanstalk**.

💡 **Use Case**: If you host a website on **CloudFront**, ACM provides **free SSL certificates**.

## ****5️⃣ Compliance & Auditing****

These tools help maintain **compliance** and **audit security configurations**.

### 🔹 **AWS Config**

✅ Tracks **configuration changes** to AWS resources.  
✅ Detects **misconfigurations** and enforces security compliance.  
✅ Supports **custom AWS Config rules** for automated security checks.

💡 **Use Case**: If an **S3 bucket is made public**, AWS Config can **trigger an alert or auto-remediation action**.

### 🔹 **AWS CloudTrail**

✅ Logs **all API calls** made in an AWS account.  
✅ Helps **investigate security incidents** and **track changes**.  
✅ Supports **real-time monitoring** with Amazon CloudWatch integration.

💡 **Use Case**: If an attacker tries to **delete an S3 bucket**, CloudTrail logs the API call for investigation.

### 🔹 **AWS Audit Manager**

✅ **Automates security compliance assessments** (CIS, PCI DSS, ISO 27001, HIPAA).  
✅ Generates **audit-ready reports** for compliance teams.  
✅ Integrates with **AWS Config and Security Hub**.

💡 **Use Case**: If your company needs to prove **PCI DSS compliance**, Audit Manager **generates reports automatically**.



# 82 Amazon S3 event notification

Amazon S3 Event Notifications allow you to receive notifications when specific events occur in an S3 bucket, such as object creation, deletion, or modification. These notifications can be sent to various AWS services, enabling automation and real-time processing.

### **Supported Event Types**

S3 can send notifications for the following events:

* **Object Created Events**
  + s3:ObjectCreated:\* – All create events
  + s3:ObjectCreated:Put – When an object is uploaded using PUT
  + s3:ObjectCreated:Post – When an object is uploaded using a POST form
  + s3:ObjectCreated:Copy – When an object is copied from another S3 location
  + s3:ObjectCreated:CompleteMultipartUpload – When a multipart upload completes
* **Object Deleted Events**
  + s3:ObjectRemoved:\* – All delete events
  + s3:ObjectRemoved:Delete – When an object is deleted
  + s3:ObjectRemoved:DeleteMarkerCreated – When a delete marker is created in versioned buckets
* **Object Restore Events** (for objects stored in Glacier or other archival storage)
  + s3:ObjectRestore:Post
  + s3:ObjectRestore:Completed
* **Other Events**
  + s3:ReducedRedundancyLostObject – When an object stored in Reduced Redundancy Storage is lost
  + s3:Replication:\* – Events related to S3 Replication

### **Destinations for Notifications**

S3 can send event notifications to:

1. **Amazon Simple Queue Service (SQS)** – To process events asynchronously using a message queue. Standard SQS queue is only allowed as an Amazon S3 event notification destination, whereas FIFO SQS queue is not allowed
2. **Amazon Simple Notification Service (SNS)** – To send notifications to multiple subscribers, such as email, SMS, or Lambda.
3. **AWS Lambda** – To trigger serverless functions for real-time processing (e.g., image processing, log analysis).

### **How to Configure S3 Event Notifications**

You can configure event notifications using:

* **AWS Management Console**
* **AWS CLI**
* **AWS SDKs**
* **Terraform or CloudFormation (for Infrastructure as Code)**

#### **Example Configuration using AWS CLI**

sh

aws s3api put-bucket-notification-configuration --bucket my-bucket --notification-configuration '{

"LambdaFunctionConfigurations": [

{

"LambdaFunctionArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",

"Events": ["s3:ObjectCreated:\*"]

}

]

}'

This setup triggers the **MyFunction** Lambda function whenever an object is created in the **my-bucket** S3 bucket.

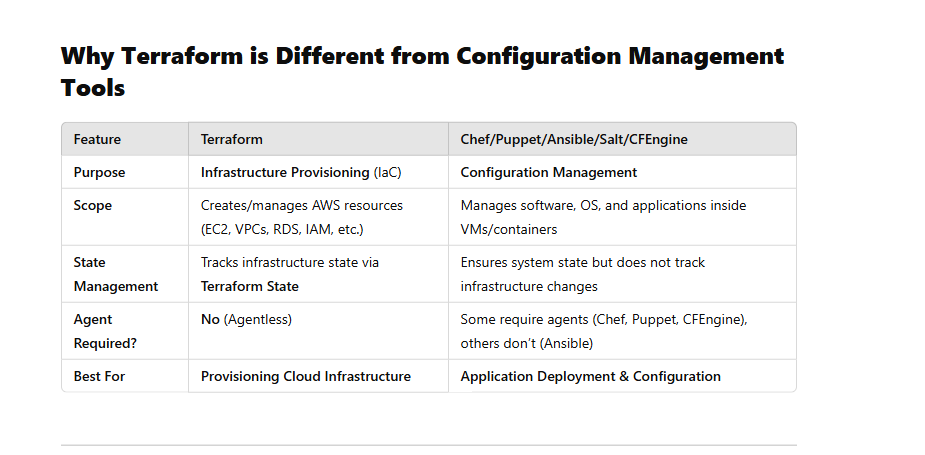
### **Use Cases**

* **Data Processing Pipelines** – Trigger Lambda for real-time processing of uploaded files.
* **Automatic Indexing** – Notify an indexing service when new documents are uploaded.
* **Event-driven Workflows** – Send notifications to SQS for processing files in batches.
* **Security & Compliance** – Detect and log unauthorized deletions.

# 83 Terraform: Infrastructure as Code (IaC) in AWS

Terraform is a powerful **Infrastructure as Code (IaC)** tool that automates the provisioning and management of cloud resources. Unlike traditional **configuration management tools** (like Chef, Puppet, Ansible, and CFEngine), Terraform is primarily used for **infrastructure provisioning** rather than system configuration.

## Why Terraform is Different from Configuration Management Tools



Terraform is widely used for **AWS infrastructure automation**, allowing DevOps teams to define cloud environments using **declarative configuration files**.

### **Key Features**

1. **Declarative Syntax (HCL - HashiCorp Configuration Language)**
   * Infrastructure is defined using **Terraform configuration files (.tf)**.
   * Example:

h

Sao chépChỉnh sửa

resource "aws\_instance" "web" {

ami = "ami-0abcdef1234567890"

instance\_type = "t2.micro"

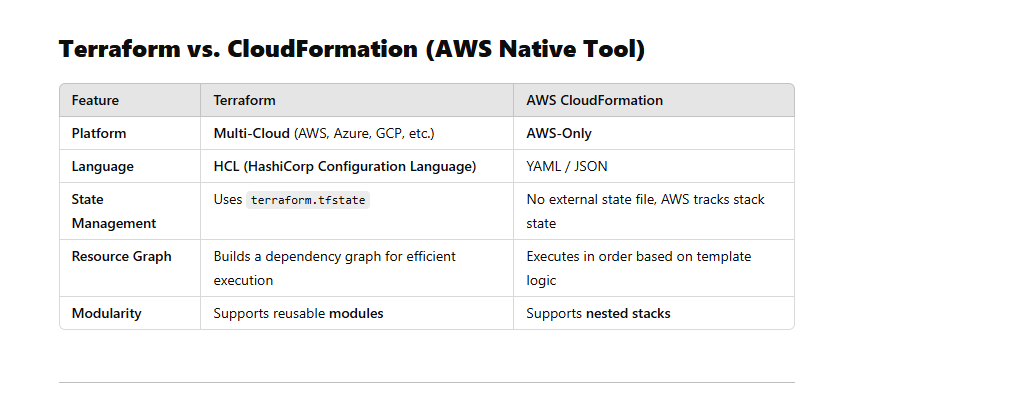
}

1. **State Management**
   * Terraform maintains an **infrastructure state file (terraform.tfstate)**, tracking changes to resources.
2. **Immutable Infrastructure**
   * Instead of modifying existing infrastructure, Terraform **replaces** resources if needed, preventing drift.
3. **Multi-Cloud Support**
   * Works with **AWS, Azure, Google Cloud, Kubernetes, and on-premise environments**.
4. **Modules & Reusability**
   * Supports **modular infrastructure**, making it easy to reuse and share configurations.

## ****Terraform Use Cases in AWS****

* **Automating AWS Resource Provisioning** (EC2, VPC, RDS, IAM, S3, Lambda, etc.).
* **Infrastructure as Code (IaC) for DevOps Pipelines**.
* **Multi-Region and Multi-Account AWS Deployments**.
* **Disaster Recovery** – Quickly recreate infrastructure in case of failure.
* **CI/CD Integration** – Works with Jenkins, GitHub Actions, and other CI/CD tools.

## Terraform vs. CloudFormation (AWS Native Tool)



# 84 what is meta data vs user data in EC2

In **AWS EC2**, there is a distinction between **metadata** and **user data**:

## 1. ****Metadata**** (Instance Metadata)

* **Definition**: Metadata is information about the EC2 instance itself, such as its instance ID, public and private IP addresses, security groups, and more.
* **Usage**: It is used by the instance to configure itself dynamically.
* **Access**: It is retrieved from within the instance using the special URL:

ruby

Sao chépChỉnh sửa

http://169.254.169.254/latest/meta-data/

* **Examples**:
  + Instance ID
  + Public and private IP addresses
  + Security group details
  + IAM role credentials
  + Availability Zone

## 2. ****User Data****

* **Definition**: User data is custom data that you provide when launching an instance. It is typically used for bootstrapping, such as running scripts to configure the instance.
* **Usage**: It is commonly used to install software, update configurations, or start services when the instance boots up.
* **Access**: It is retrieved from:

ruby

Sao chépChỉnh sửa

http://169.254.169.254/latest/user-data/

* **Examples**:
  + Shell script to install software (yum install -y httpd)
  + Cloud-init directives to configure the instance
  + Custom application configuration

## Key Differences



# 85 What happen? when i terminal EC2 attached on EBS.

When you **terminate an EC2 instance**, what happens to the **attached EBS volumes** depends on how they were configured:

### **1️⃣ Root EBS Volume (Default Behavior)**

* By default, the root EBS volume is **deleted** when the EC2 instance is terminated.
* This is controlled by the **"Delete on Termination"** flag.

### **2️⃣ Additional (Non-root) EBS Volumes**

* Additional EBS volumes **are NOT deleted** by default when an EC2 instance is terminated.
* They remain in your AWS account and can be **attached to another instance**.
* You can change this behavior using the **Delete on Termination** setting.

### **How to Prevent EBS Deletion?**

* When launching an EC2 instance, modify the root volume settings:
  + **AWS Console:** Uncheck the **"Delete on Termination"** option.
  + **AWS CLI:** Use --no-delete-on-termination.

#### **Check Delete on Termination Setting (CLI)**

sh

Sao chépChỉnh sửa

aws ec2 describe-volumes --volume-ids vol-xxxxxxxx

#### **Modify to Keep EBS After Termination**

sh

Sao chépChỉnh sửa

aws ec2 modify-instance-attribute --instance-id i-xxxxxxxx --block-device-mappings "[{\"DeviceName\"

# 86 Kinesis Agent (work)

Kinesis Agent

Kinesis Agent cannot write to a Kinesis Firehose for which the delivery stream source is already set as Kinesis Data Streams?

If the **Kinesis Agent** cannot write to an **Amazon Kinesis Firehose** delivery stream where the **source is set as Kinesis Data Streams**, it's because **Firehose does not allow direct ingestion from Kinesis Agent when it is configured to use a Kinesis Data Stream as its source**. Here’s why and how you can resolve it:

### **Why This Happens**

* **Firehose supports two types of sources:**
  1. **Direct PUT (default source)** – Allows Kinesis Agent, SDK, or AWS CLI to write directly to Firehose.
  2. **Kinesis Data Stream as a source** – Firehose pulls data from an existing Kinesis Data Stream instead of receiving direct PUTs.
* If Firehose is configured to use **Kinesis Data Stream as its source**, it **only** pulls data from that stream, and direct PUTs (such as from Kinesis Agent) are **not allowed**.

# 87 What is Resource Access Manager

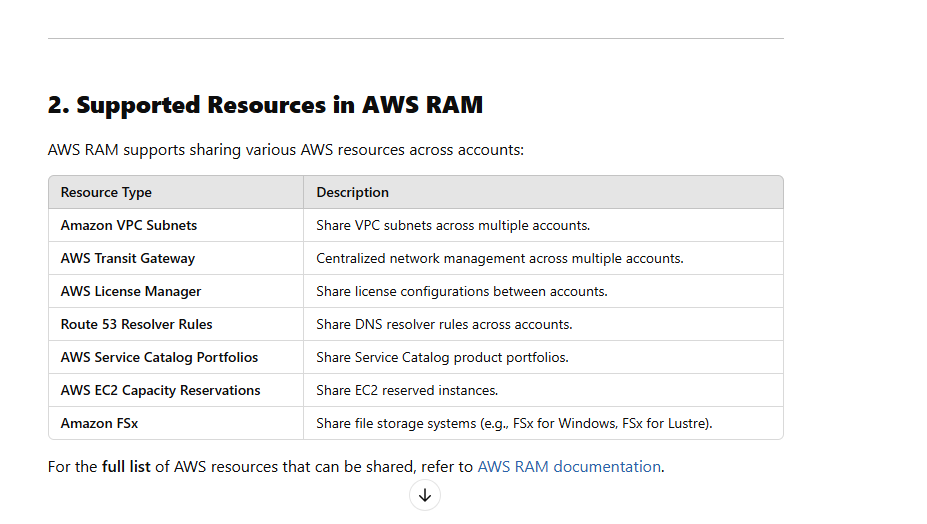
AWS **Resource Access Manager (RAM)** is a service that allows you to securely share AWS resources across different AWS accounts or within an AWS Organization. This helps you avoid resource duplication and centralize management.

### **Key Features of AWS RAM**

1. **Cross-Account Sharing** – Share resources with specific AWS accounts or across an AWS Organization.
2. **Supports Multiple Resource Types** – Shareable resources include VPC subnets, Transit Gateways, Route 53 Resolver rules, License Manager configurations, etc.
3. **Security and Access Control** – Uses AWS Identity and Access Management (IAM) for permissions.
4. **Integration with AWS Organizations** – Enables easier sharing within organizational units (OUs).
5. **Reduces Costs** – Instead of creating duplicate resources, you can share existing ones.

### **Common Use Cases**

* Sharing **VPC subnets** across accounts to enable networking without duplication.
* Sharing **AWS Transit Gateway** to centralize network management.
* Sharing **Route 53 Resolver rules** to maintain a unified DNS resolution strategy.



No, **AWS Resource Access Manager (AWS RAM) can be used both within and outside an AWS Organization**. However, the way you share resources depends on whether you are using AWS Organizations or standalone AWS accounts.

# 88 What is EC2 Dedicated Hosts

Amazon EC2 **Dedicated Hosts** are physical servers fully dedicated to your use. Unlike regular EC2 instances that run on shared hardware, Dedicated Hosts give you more control over instance placement, licensing, and compliance.

### **Key Features of EC2 Dedicated Hosts**

1. **Full Server Control** – You get a dedicated physical server where you can launch EC2 instances.
2. **Licensing Management** – Helps with Bring Your Own License (BYOL) for software like Windows Server, SQL Server, and Oracle.
3. **Cost Savings for Long-Term Usage** – You can save money if you commit to a **Dedicated Host Reservation**.
4. **Compliance & Security** – Useful for organizations with strict regulatory requirements (e.g., HIPAA, GDPR).
5. **Instance Affinity** – Ensures instances are always placed on the same host, preventing unexpected migrations.
6. **Visibility into Hardware** – You can track instance placement and optimize resource allocation.

### **When to Use Dedicated Hosts?**

* **Regulatory Compliance:** If your workload requires isolation for compliance reasons.
* **Software Licensing:** If you need to use existing per-core or per-socket software licenses.
* **Performance & Security:** When you want full control over hardware for security-sensitive applications.

# 89 AWS Spot Fleet

A **Spot Fleet** in AWS is a collection of **Spot Instances** (and optionally **On-Demand Instances**) that AWS automatically manages to meet your specified capacity and price requirements. It helps optimize costs while ensuring high availability for workloads that can tolerate interruptions.

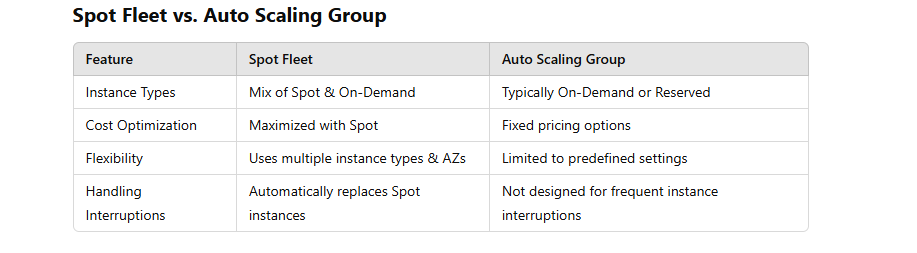
### **Key Features**

1. **Cost Optimization** – Spot Instances can be up to **90% cheaper** than On-Demand.
2. **Auto Scaling** – Automatically adjusts capacity based on availability and pricing.
3. **Diverse Instance Types** – You can request different instance types across multiple Availability Zones.
4. **Fallback to On-Demand** – Ensures availability by including On-Demand instances when Spot capacity is unavailable.
5. **Capacity Pools** – AWS selects the best-priced instances from multiple Spot markets.
6. **Fleet Management** – AWS maintains the requested capacity by replacing interrupted Spot Instances.

### **Use Cases**

* **Big Data & Batch Processing** – Cost-efficient for jobs that can be interrupted and resumed.
* **Containerized Applications** – Works well with Kubernetes, ECS, and EKS.
* **Machine Learning & AI Training** – Reduces costs for large-scale training jobs.
* **Rendering & Simulations** – Ideal for workloads that require large-scale compute power.

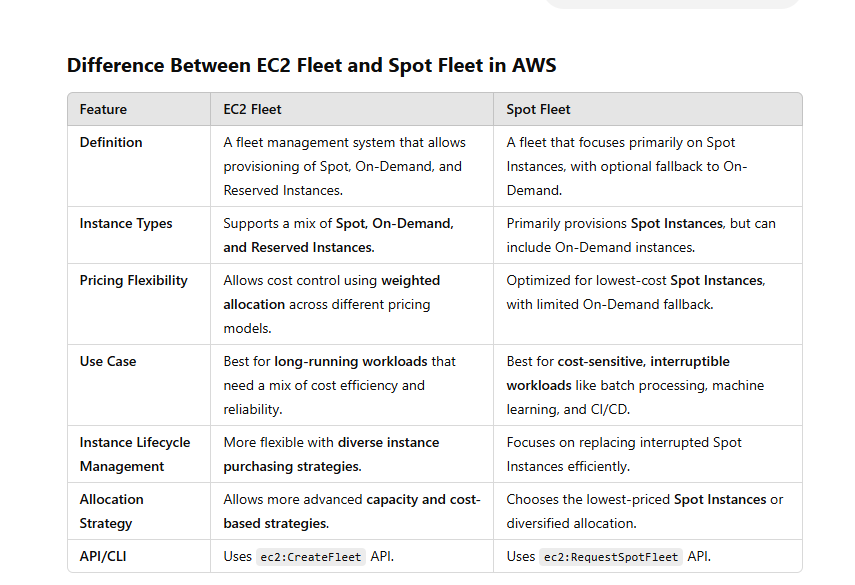
### **Spot Fleet vs. Auto Scaling Group**



### **How to Create a Spot Fleet**

1. Define **Instance Types, Capacity, and Price Limits**.
2. Choose **Allocation Strategy** (e.g., lowest price, diversified, capacity-optimized).
3. Configure **IAM Role** for managing instances.
4. Submit the Spot Fleet request.

# 90 Difference Between EC2 Fleet and Spot Fleet in AWS



### **Key Takeaways**

* **EC2 Fleet** → More flexible, supports Spot, On-Demand, and Reserved Instances.
* **Spot Fleet** → Optimized for Spot pricing, with optional On-Demand fallback.
* **Use EC2 Fleet** if you need balanced cost and reliability.
* **Use Spot Fleet** if you want to maximize cost savings and can handle interruptions.

Use case using

## 🚀 ****EC2 Fleet – Use Cases****

💡 When you want **flexibility**, **cost optimization**, and **custom control** over your instance fleet.

### ✅ **Use Cases**:

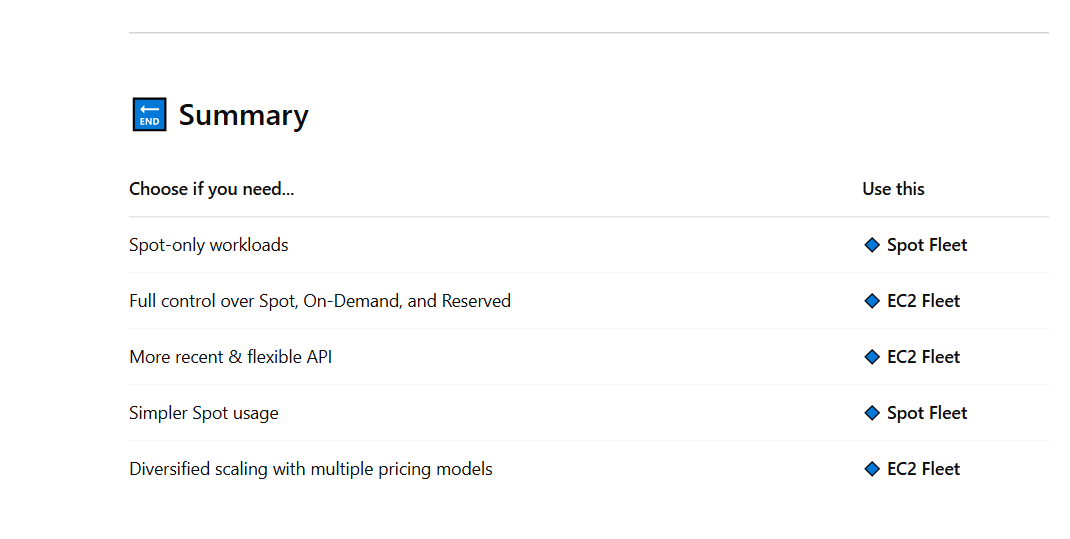
1. **High-Performance Computing (HPC)** or **Big Data Processing**
   * Example: Running genomics pipelines, simulations, data lakes.
   * Why: EC2 Fleet lets you combine Spot for cost savings, On-Demand for stability, and Reserved Instances for predictability.
2. **Burst Workloads or Batch Jobs**
   * Example: Video rendering, report generation, ETL jobs.
   * Why: Use EC2 Fleet to get fast allocation across Spot/On-Demand based on your cost or speed priorities.
3. **CI/CD Build Farms**
   * Example: Jenkins agents for builds/tests.
   * Why: Use cheap Spot instances with fallback to On-Demand so jobs don’t get blocked.
4. **Massive Scaling with Control Over Pricing Models**
   * Example: Launch 500 instances with 60% Spot, 30% On-Demand, 10% Reserved.
   * Why: EC2 Fleet supports that precise control.

## 🌩️ ****Spot Fleet – Use Cases****

💡 When your workload is **Spot-only or Spot-heavy** and you want to **keep it simple**.

### ✅ **Use Cases**:

1. **Cost-Sensitive Batch Jobs (Spot-only)**
   * Example: Machine learning training with retry logic.
   * Why: Spot Fleet is simple, optimized for Spot and lets AWS pick the best pool.
2. **Stateless Web Services or Containers**
   * Example: Services in ECS/EKS using Spot.
   * Why: Use Spot Fleet to fill capacity across AZs and instance types while saving money.
3. **Basic Scaling Without Full Flex Control**
   * Example: Migrate legacy job runners from On-Demand to Spot.
   * Why: Spot Fleet is focused and easy to integrate with existing tools.



# 91 what is EC2 hibernate

EC2 Hibernate is a feature in **Amazon EC2 (Elastic Compute Cloud)** that allows you to pause and resume an instance while maintaining its in-memory state. When you hibernate an EC2 instance:

1. **RAM contents are saved** to the instance’s root volume (EBS).
2. **Instance state is preserved** (unlike stopping, which requires a fresh boot).
3. **Instance ID, private/public IPs remain unchanged** when resumed.
4. **Billing is paused** for compute, but storage costs continue.

### **How It Works**

* When you hibernate an instance, EC2 signals the OS to save RAM to disk (EBS).
* The instance is then stopped, preserving its RAM and other configurations.
* When you restart, the saved memory state is restored, skipping OS boot time.

### **Use Cases**

* Quickly resuming long-running workloads.
* Avoiding reloading large applications into memory.
* Preserving session data for applications with long startup times.

### **Supported Instances**

* Only **Amazon EBS-backed** instances.
* **Supported instance families**: C, M, R, T, and X series with certain limitations.
* **RAM limit**: 150GB max.
* **Operating Systems**: Amazon Linux, Ubuntu, Windows, and RHEL with proper hibernation support.

# 92 What happens when you stop an instance

###### Changes registered at the OS level

* The API request sends a button press event to the guest.
* Various system services are stopped as a result of the button press event. Graceful shutdown is triggered by the ACPI shutdown button press event from the hypervisor.
* ACPI shutdown is initiated.
* The instance shuts down when the graceful shutdown process exits. There is no configurable OS shutdown time.
* If the instance OS does not cleanly shut down within a few minutes, a hard shutdown is performed.
* The instance stops running.
* The instance status changes to stopping and then stopped.
* [Auto Scaling] If your instance is in an Auto Scaling group, when the instance is in any Amazon EC2 state other than running, or if its status for the status checks becomes impaired, Amazon EC2 Auto Scaling considers the instance to be unhealthy and replaces it. For more information, see [Health checks for instances in an Auto Scaling group](https://docs.aws.amazon.com/autoscaling/ec2/userguide/ec2-auto-scaling-health-checks.html) in the Amazon EC2 Auto Scaling User Guide.
* [Windows instances] When you stop and start a Windows instance, the launch agent performs tasks on the instance, such as changing the drive letters for any attached Amazon EBS volumes. For more information about these defaults and how you can change them, see [Use the EC2Launch v2 agent to perform tasks during EC2 Windows instance launch](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2launch-v2.html).

###### Resources lost

* Data stored on the RAM.
* Data stored on the instance store volumes.
* The public IPv4 address that Amazon EC2 automatically assigned to the instance upon launch or start. To retain a public IPv4 address that never changes, you can associate an [Elastic IP address](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html) with your instance.

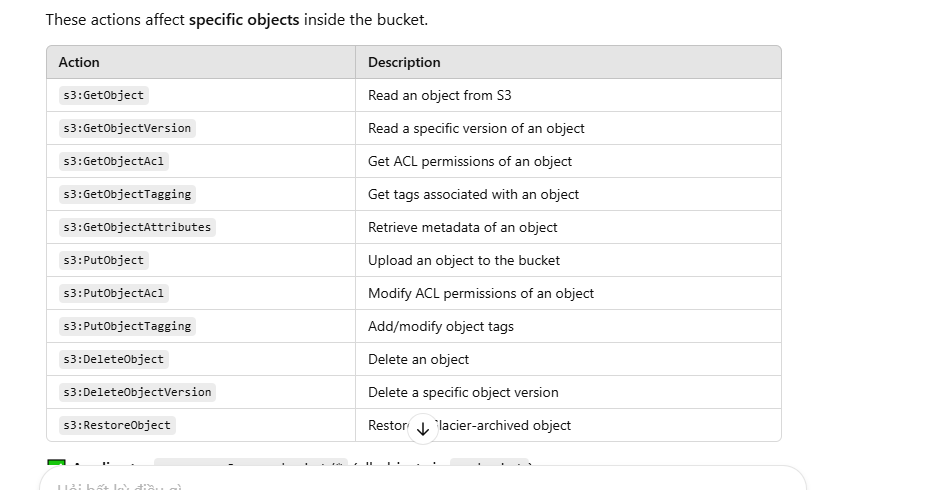
###### Resources that persist

* Any attached Amazon EBS volumes.
* Data stored on the attached Amazon EBS volumes.
* Private IPv4 addresses.
* IPv6 addresses.
* Elastic IP addresses associated with the instance. Note that when the instance is stopped, you are [charged for the associated Elastic IP](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html#eip-pricing)

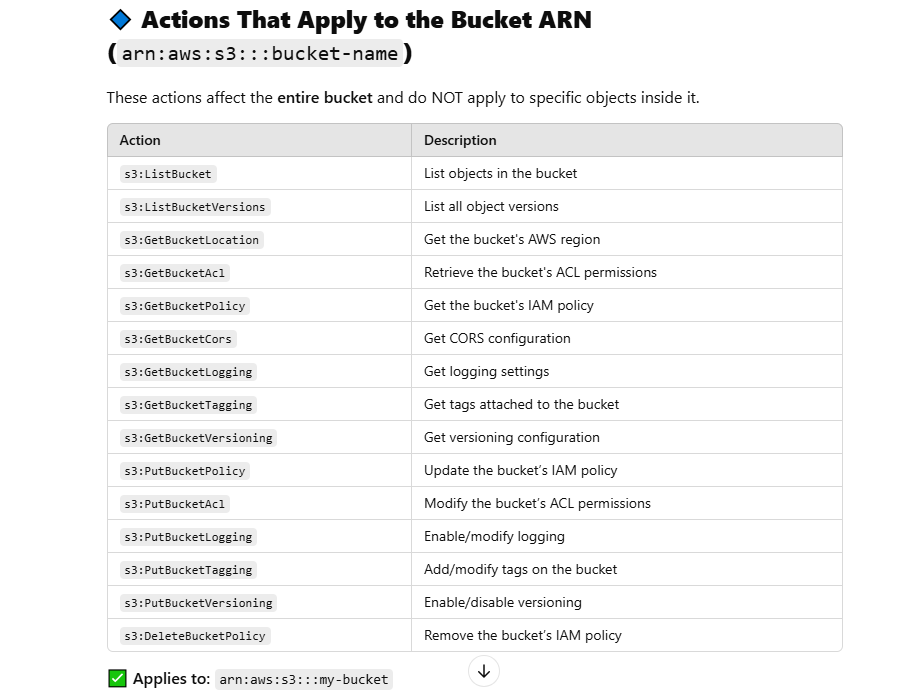
# 93****How to Decide Which ARN to Use?****

* If the action **modifies or retrieves bucket settings** → **Use bucket ARN (arn:aws:s3:::bucket-name)**
* If the action **reads, writes, or deletes specific objects** → **Use object ARN (arn:aws:s3:::bucket-name/\*)**

## **Actions That Apply to Object ARNs (**arn:aws:s3:::bucket-name/\***)**



## Actions That Apply to the Bucket ARN (arn:aws:s3:::bucket-name)



# 94 what is DynamoDB with Provisioned Capacity

Amazon DynamoDB with **Provisioned Capacity** is a mode where you manually specify the amount of **read** and **write** capacity units (RCUs and WCUs) that your application needs. This mode is ideal when you have predictable traffic patterns and want cost optimization by reserving throughput rather than paying per request.

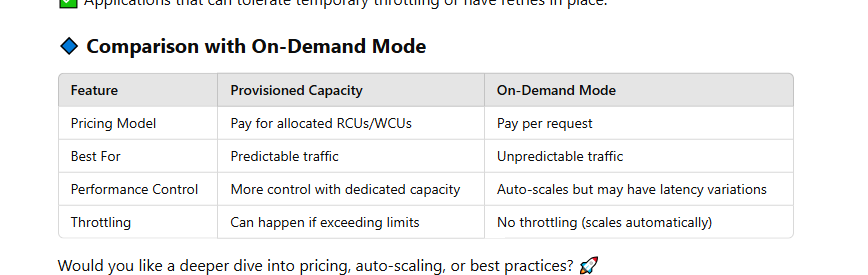
### 🔹 **How It Works**

* You allocate a fixed number of RCUs and WCUs.
* If your workload stays within this limit, you get **consistent performance** at a lower cost.
* If your workload exceeds the provisioned limit, requests can be **throttled** unless you enable **Auto Scaling** to adjust capacity automatically.

### 🔹 **When to Use It?**

✅ Workloads with **predictable traffic** (e.g., steady database usage).  
✅ Cost-sensitive applications where **reserved capacity is cheaper** than on-demand.  
✅ Applications that can tolerate temporary throttling or have retries in place.

### 🔹 **Comparison with On-Demand Mode**



### **Read Capacity Units (RCUs)**

* **1 RCU** = **1 strongly consistent read of 4 KB per second**
* **1 RCU** = **2 eventually consistent reads of 4 KB per second**
* **1 RCU** = **1 transactional read of 4 KB per second** (costs **2x** more)

👉 If your item size is **8 KB**, it will consume **2 RCUs** for a strongly consistent read.

### **Write Capacity Units (WCUs)**

* **1 WCU** = **1 write of 1 KB per second**
* **Transactional writes** cost **2x** the WCU of a normal write.

👉 If your item size is **2 KB**, it will consume **2 WCUs**.

# 94 Amazon Aurora Global Databases

### **🔹 Amazon Aurora Global Databases**

Amazon **Aurora Global Database** is a feature that allows a **single Aurora database** to span **multiple AWS regions**, providing **low-latency global reads** and **disaster recovery**.

### **🔹 How It Works**

1. **One Primary Region for Writes**
   * The **primary region** handles **all write operations**.
   * It replicates data to **up to 5 secondary AWS regions**.
2. **Cross-Region Read Replicas** (Low Latency Reads)
   * Each secondary region gets **read replicas** that support **fast, local reads** (latency < 1 second).
   * Applications in different regions can read from the closest replica instead of the primary region.
3. **Fast Cross-Region Replication**
   * Uses **Aurora’s dedicated replication technology**, replicating **in milliseconds**.
   * Unlike traditional **RDS cross-region replication**, which is **slower and uses binlogs**, Aurora replication is nearly **real-time**.
4. **Disaster Recovery (Failover)**
   * If the **primary region fails**, a **secondary region can be promoted** to the new primary **within 1 minute**.

### **🔹 Key Benefits**

✅ **Low Latency Reads** – Users worldwide get fast access to data.  
✅ **High Availability & DR** – Automatic failover in case of region failure.  
✅ **Global Scaling** – Up to **16 read replicas per secondary region**.  
✅ **Fast Replication** – Cross-region replication latency is **< 1 second**.  
✅ **Multi-Region Support** – Supports **up to 5 secondary regions**.

### **🔹 Use Cases**

🔹 **Global Applications** – Apps serving users across multiple continents.  
🔹 **Disaster Recovery** – Ensures availability if a region goes down.  
🔹 **Regulatory Compliance** – Store customer data in specific regions.  
🔹 **Gaming & Finance** – Handle real-time, high-read workloads globally.

# 95 URL of Bucket

## Static website

http://<your-bucket-name>.s3-website-<region>.amazonaws.com

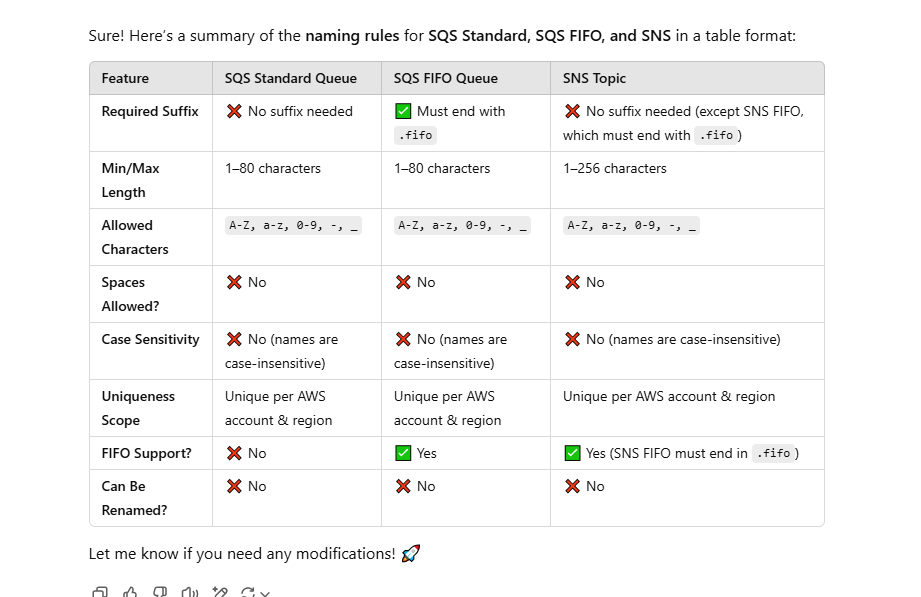
http://<your-bucket-name>.s3-website.<region>.amazonaws.com

## url bucket

https://<bucket-name>.s3.<region>.amazonaws.com/<object-key>

https://s3.<region>.amazonaws.com/<bucket-name>/<object-key>

# 96 **naming rules** for **SQS Standard, SQS FIFO, and SNS** in a table format:



# 97 what is DataSync in aws

 AWS DayaSync supports only NFS and SMB file types

AWS DataSync is a managed service that automates, accelerates, and simplifies moving large amounts of data between on-premises storage, AWS storage services, and even between AWS services. It is optimized for **fast, secure, and automated data transfers**.

### **Key Features of AWS DataSync**

1. **Automated Data Transfers** – Supports scheduling and incremental syncs.
2. **High-Speed Transfer** – Uses a purpose-built protocol that speeds up data movement compared to traditional methods.
3. **Data Integrity & Security** – Encrypts data in transit and verifies integrity during transfers.
4. **Supports Multiple Storage Services**:
   * Amazon S3
   * Amazon EFS (Elastic File System)
   * Amazon FSx (for Windows and Lustre)
   * Amazon S3 Glacier
   * Amazon S3-compatible storage
   * On-premises NFS and SMB storage

### **Common Use Cases**

* **Data Migration** – Move data from on-premises to AWS storage.
* **Backup & Disaster Recovery** – Automate regular syncs to AWS for backup.
* **Hybrid Cloud Workflows** – Sync data between AWS and on-prem systems.
* **Data Processing Pipelines** – Transfer large datasets efficiently for analytics.

# 98 what is VPC console wizard

The **VPC Console Wizard** in AWS is a guided tool in the AWS Management Console that helps you quickly create and configure **Amazon Virtual Private Cloud (VPC)** components. Instead of manually setting up each networking resource (such as subnets, route tables, and internet gateways), the wizard provides pre-configured options to streamline the process.

### **Options in the VPC Console Wizard**

When creating a VPC, the wizard offers several setup choices, including:

1. **VPC with a Single Public Subnet**
   * Creates a VPC with a single public subnet.
   * Includes an Internet Gateway for public internet access.
   * Useful for applications that need to be publicly accessible.
2. **VPC with Public and Private Subnets**
   * Creates both public and private subnets.
   * The public subnet connects to the internet via an Internet Gateway.
   * The private subnet can access the internet through a **NAT Gateway**.
   * Suitable for applications where only certain resources should be public.
3. **VPC with Public and Private Subnets and Hardware VPN Access**
   * Similar to the previous setup but adds a **VPN** connection to securely connect on-premises networks to AWS.
4. **VPC with Private Subnets and AWS Site-to-Site VPN Access**
   * Creates a private-only VPC without direct internet access.
   * Connects to an on-premises network using a **VPN**.
   * Ideal for private workloads that shouldn't be exposed to the public internet.

### **Benefits of Using the VPC Console Wizard**

* **Simplifies network setup** by pre-configuring subnets, route tables, and gateways.
* **Reduces manual errors** when setting up networking components.
* **Speeds up deployment** for AWS resources requiring a VPC.

# 99 What is AWS **Resource Access Manager (RAM)**

AWS **Resource Access Manager (RAM)** is a service that allows you to securely share AWS resources across different AWS accounts or within an AWS Organization. This helps you avoid resource duplication and centralize management.

### **Key Features of AWS RAM**

1. **Cross-Account Sharing** – Share resources with specific AWS accounts or across an AWS Organization.
2. **Supports Multiple Resource Types** – Shareable resources include VPC subnets, Transit Gateways, Route 53 Resolver rules, License Manager configurations, etc.
3. **Security and Access Control** – Uses AWS Identity and Access Management (IAM) for permissions.
4. **Integration with AWS Organizations** – Enables easier sharing within organizational units (OUs).
5. **Reduces Costs** – Instead of creating duplicate resources, you can share existing ones.

### **Common Use Cases**

* Sharing **VPC subnets** across accounts to enable networking without duplication.
* Sharing **AWS Transit Gateway** to centralize network management.
* Sharing **Route 53 Resolver rules** to maintain a unified DNS resolution strategy.

# 100 what is IAM permission boundary

### **IAM Permission Boundary in AWS**

An **IAM permission boundary** is an advanced security feature in AWS Identity and Access Management (IAM) that **defines the maximum permissions an IAM entity (user or role) can have**. It acts as an **upper limit** for the permissions that the entity can use.

### **How It Works**

* Normally, an IAM **policy** grants permissions to a user or role.
* A **permission boundary** **restricts** those permissions, even if a policy grants more.
* The **effective permissions** of an entity are the **intersection** of the permissions granted by IAM policies and the permissions allowed by the boundary.

### **Use Case Example**

1. You create an IAM **role** with a policy that allows s3:\* (full S3 access).
2. You attach a **permission boundary** that only allows s3:ListBucket.
3. **Result**: Even though the role's policy allows full access to S3, the permission boundary limits it to **only listing** buckets.

### **Why Use Permission Boundaries?**

* **Security Control**: Limits what IAM users or roles can do, preventing excessive permissions.
* **Delegated Administration**: Allows teams to create IAM roles while ensuring they don’t exceed defined limits.
* **Multi-Team Environments**: Ensures that developers or admins don’t over-provision themselves.

### **Key Points**

* **Only applies to IAM users and roles** (not groups).
* **Does not grant permissions**—it only restricts them.
* **Works with other IAM policies** (identity policies, resource policies, SCPs).
* **Applies to session policies** in AWS STS.

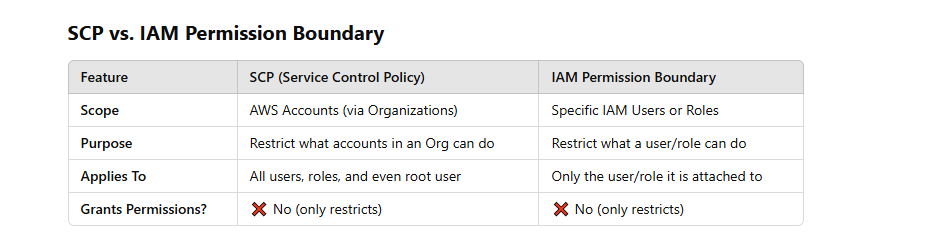
# 101 Service Control Policy (SCP) in AWS

A **Service Control Policy (SCP)** is a policy in **AWS Organizations** that **sets permission guardrails** at the **organization, organizational unit (OU), or account level**. It defines the **maximum permissions** that accounts within an AWS Organization can have.

### **How SCP Works**

* SCPs **do not grant permissions**; they only **restrict** what AWS IAM policies can do.
* If an action is **not allowed** by an SCP, IAM users and roles **within the account cannot perform** it—even if an IAM policy grants the permission.
* SCPs apply to **all IAM users, groups, and roles**, including the **root user** of an AWS account.

### **SCP vs. IAM Permission Boundary**



# 102 EC2 Fleet vs. Instance Fleet

Both **EC2 Fleet** and **Instance Fleet** are AWS features designed for managing multiple EC2 instances efficiently, but they serve different purposes. Here's a breakdown:

### **1. EC2 Fleet**

**EC2 Fleet** is a broader AWS feature that allows you to launch and manage a mix of **On-Demand, Reserved, and Spot Instances** across multiple **instance types** and **Availability Zones (AZs)**. It optimizes cost and capacity for a variety of workloads.

#### **Key Features:**

* Supports **multiple instance types** across different Availability Zones.
* Can launch a mix of **On-Demand, Spot, and Reserved Instances**.
* Uses **Weighted Capacity Allocation**, allowing you to prioritize specific instance types.
* Works **independently of AWS EMR** (i.e., it’s for general EC2 workloads).

#### **Use Cases:**

* High Availability and Fault Tolerance across AZs.
* **Batch processing**, machine learning, or web applications requiring flexible compute.
* Cost-optimized scaling using Spot Instances.

### **2. Instance Fleet (Specific to AWS EMR)**

**Instance Fleet** is an AWS **EMR-specific** feature that allows you to specify a mix of **instance types and purchase options** for EMR clusters. It helps balance cost and availability for **big data workloads**.

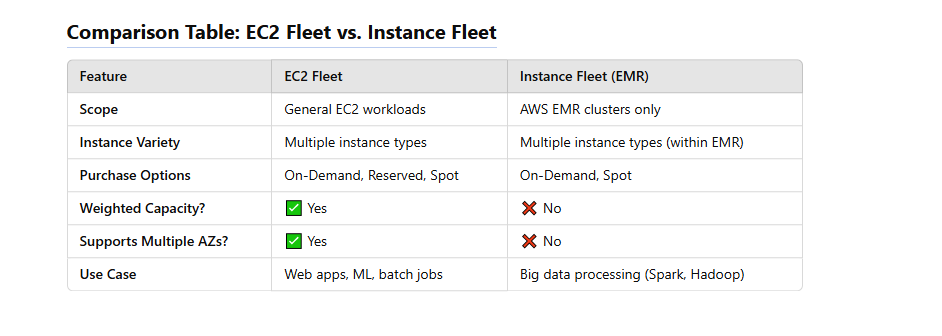
#### **Key Features:**

* **Specific to AWS EMR** (not general EC2 usage).
* Replaces the older **Instance Group model** in EMR.
* Allows **diverse instance types** within each fleet (Master, Core, Task nodes).
* Uses **Spot and On-Demand mix** to optimize cost.
* Does **not use Weighted Capacity Allocation** like EC2 Fleet.

#### **Use Cases:**

* **Big data processing** using Apache Spark, Hadoop, or Presto.
* **Flexible scaling** of EMR clusters while controlling cost.
* **Workloads that can tolerate interruptions** (Spot Instances).

Comparison Table: EC2 Fleet vs. Instance Fleet



# 103 Dedicated Host

A **Dedicated Host** is a **physical server** that is fully dedicated to a single customer. It allows more control over **hardware affinity**, compliance, and licensing requirements.

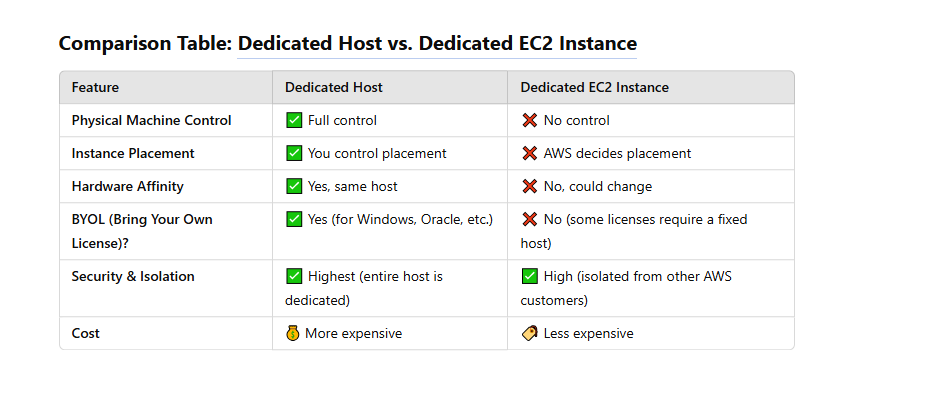
#### **Key Features:**

* Provides full control over instance placement on the host.
* Ensures that no other customers share the underlying physical hardware.
* Useful for **Bring Your Own License (BYOL)** scenarios for Windows or Oracle workloads.
* Helps meet **regulatory and compliance** requirements.
* More expensive than shared instances but offers greater isolation.

#### **Use Cases:**

* Compliance-driven industries (e.g., finance, healthcare).
* Workloads requiring **specific hardware affinity**.
* BYOL software that requires dedicated infrastructure.

## Dedicated Host vs. Dedicated EC2 Instance



# 104 NAT Instance VS NAT GATEWAY

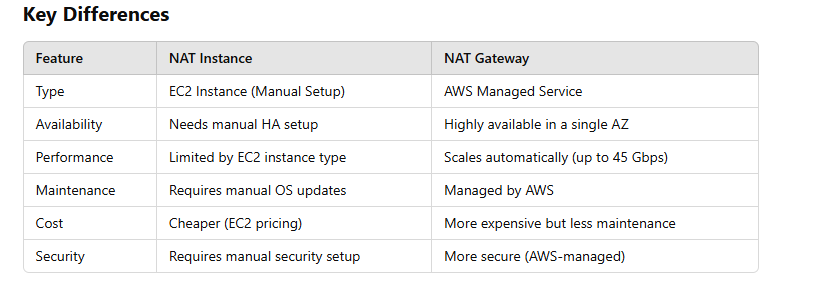
### **1. NAT Instance**

* A **NAT Instance** is an **EC2 instance** that acts as a Network Address Translation (NAT) device.
* You manually launch an EC2 instance and configure it as a NAT.
* It allows outbound internet traffic from instances in a **private subnet**.
* You have full control over it, meaning you can choose the instance type, apply security groups, and configure custom routing.
* It requires **manual configuration** for high availability (e.g., setting up multiple instances with failover).
* Performance is limited by the EC2 instance type.
* Needs manual updates for OS patches and security fixes.

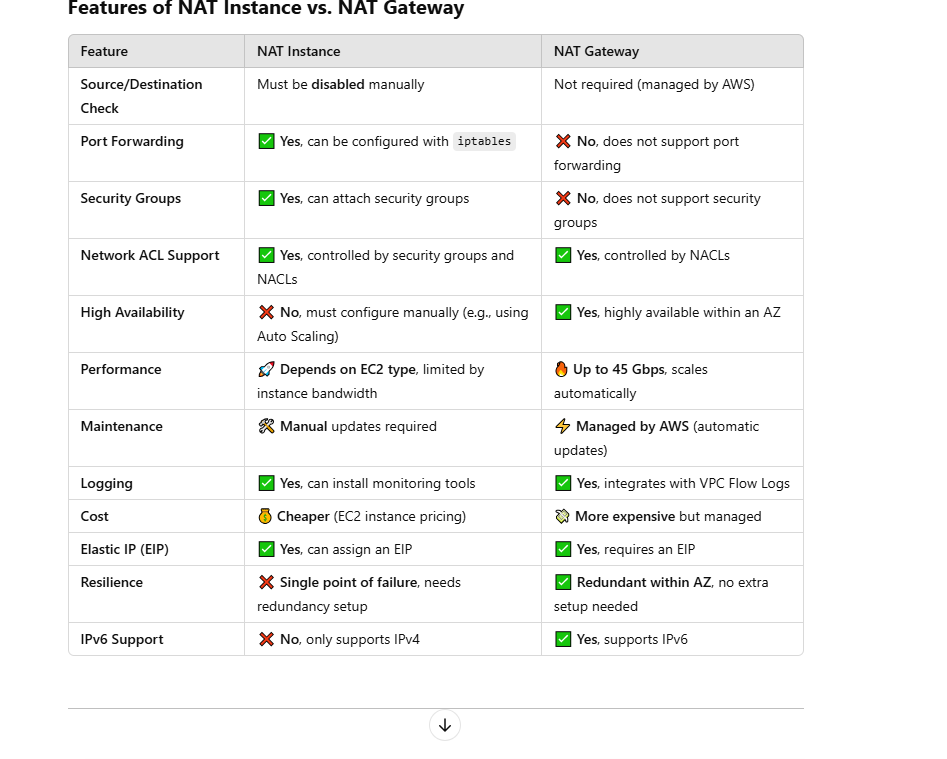
### **2. NAT Gateway**

* A **NAT Gateway** is a **managed AWS service** that provides NAT functionality.
* It is **highly available** within a single Availability Zone.
* Supports **automatic scaling** and provides higher throughput (up to **45 Gbps**).
* AWS handles maintenance, updates, and redundancy.
* More **secure** as it integrates with AWS services like VPC Flow Logs.
* More **expensive** than a NAT instance but requires less management.

## Key Differences



## Features of NAT Instance vs. NAT Gateway



### **Key Differences**

* **Port Forwarding:** Only **NAT Instances** allow port forwarding (via iptables), but **NAT Gateways do not**.
* **High Availability:** NAT Gateway is **automatically redundant**, but NAT Instances require **manual setup**.
* **Performance:** NAT Gateway can handle **higher throughput (up to 45 Gbps)**, while NAT Instances depend on **EC2 instance type**.
* **Security:** NAT Gateway is **more secure** (no SSH access, no security group config needed).

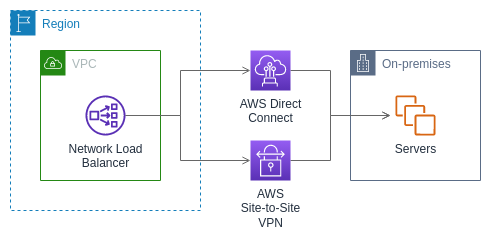
# 105 Network loadbalaning (Request routing and IP addresses)

If you specify targets using an instance ID, traffic is routed to instances using the primary private IP address that is specified in the primary network interface for the instance. The load balancer rewrites the destination IP address from the data packet before forwarding it to the target instance.

If you specify targets using IP addresses, you can route traffic to an instance using any private IP address from one or more network interfaces. This enables multiple applications on an instance to use the same port. Note that each network interface can have its own security group. The load balancer rewrites the destination IP address before forwarding it to the target.

### On premises resources as targets

On premises resources linked through AWS Direct Connect or a Site-to-Site VPN connection can serve as a target, when the target type is ip.



When using on premises resources, the IP addresses of these targets must still come from one of the following CIDR blocks:

* 10.0.0.0/8 ([RFC 1918](https://tools.ietf.org/html/rfc1918))
* 100.64.0.0/10 ([RFC 6598](https://tools.ietf.org/html/rfc6598))
* 172.16.0.0/12 (RFC 1918)
* 192.168.0.0/16 (RFC 1918)

# 106 what is warm pool. in asg

In **AWS Auto Scaling Groups (ASG)**, a **Warm Pool** is a group of pre-initialized EC2 instances that are kept in a stopped or running state to improve the scaling response time. It helps reduce the launch time of new instances when scaling out.

### **Key Features of Warm Pools in ASG**

1. **Faster Scaling:** Instead of launching new instances from scratch, ASG can quickly start stopped instances from the warm pool, reducing boot time.
2. **Cost Efficiency:** Instances in the warm pool can be in a stopped state, reducing costs compared to running instances.
3. **Pre-Initialization:** You can run startup scripts on these instances before they enter the warm pool, ensuring they are ready to handle traffic quickly.
4. **Instance Lifecycle States:** Instances in the warm pool can be in one of the following states:
   * **Stopped:** Default state, where instances are kept without running costs for compute.
   * **Running:** Instances remain running but idle, ready for immediate use.
   * **Hibernated:** Instances with hibernation enabled can be resumed quickly while retaining memory.

### **How it Works**

* When scaling out, ASG first pulls instances from the warm pool.
* If the warm pool is empty, it launches new instances from scratch.
* When scaling in, instead of terminating instances, ASG moves them to the warm pool for future use.

### **Use Case**

* Applications with bursty workloads that require quick response times.
* Workloads with expensive startup times (e.g., Java applications with long initialization).
* Cost-sensitive environments where keeping idle instances running is too expensive.

# 107 what is VPC sharing

**VPC Sharing** is a feature in AWS that allows multiple AWS accounts to share a **single Amazon Virtual Private Cloud (VPC)**. Instead of each account creating its own VPC, one account (the **owner account**) creates and manages the VPC, while other accounts (the **participant accounts**) can use its subnets.

### **Key Features of VPC Sharing**

1. **Centralized VPC Management** – The owner account manages networking resources, reducing overhead for participant accounts.
2. **Subnets are Shared, Not the Whole VPC** – Participants can use designated subnets but **cannot manage the VPC itself**.
3. **Reduced Cost & Complexity** – Avoids duplicating networking components (e.g., NAT gateways, VPC peering).
4. **Security & Control** – The owner account maintains control over network configurations like **security groups, route tables, and VPC flow logs**.
5. **Inter-Account Communication** – Resources within the shared VPC can communicate directly without needing VPC peering.

### **How VPC Sharing Works**

* The **owner account**:
  + Creates the VPC.
  + Shares specific subnets with participant accounts using **AWS Resource Access Manager (RAM)**.
* The **participant account**:
  + Launches resources (e.g., EC2, RDS) in the shared subnets.
  + Uses the networking infrastructure managed by the owner account.

### **Use Case Scenarios**

* **Centralized Networking**: A single team manages networking while different teams deploy resources.
* **Multi-Account Architecture**: Different AWS accounts (e.g., Dev, QA, Production) use a shared VPC instead of creating separate VPCs.
* **Cost Optimization**: Shared infrastructure reduces costs for NAT gateways, transit gateways, and inter-VPC traffic.

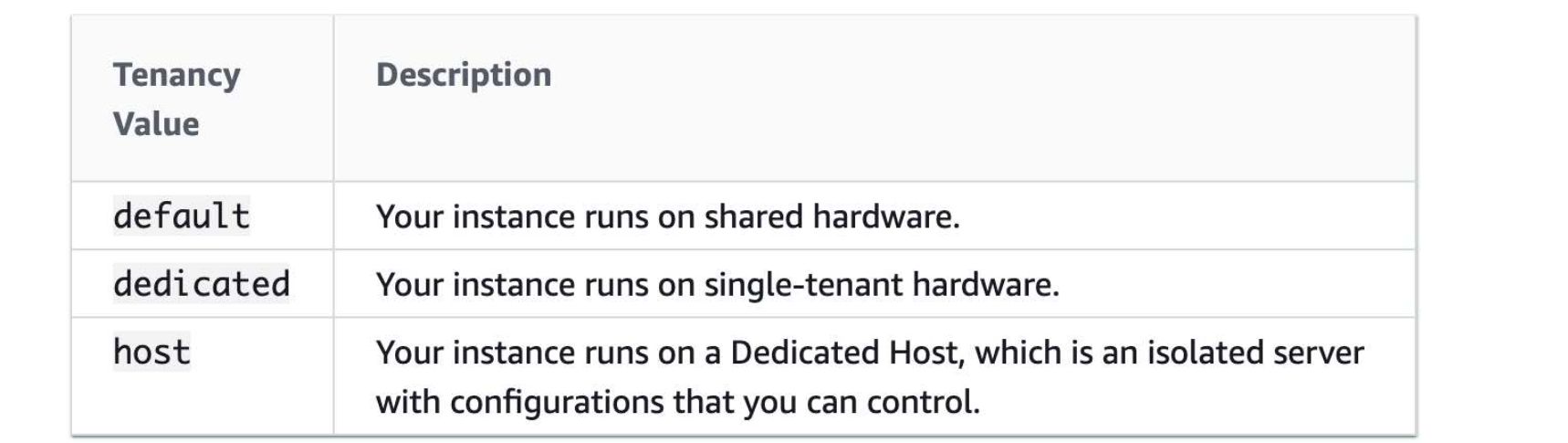
# 108 Change the instance tenancy of a VPC

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/dedicatedinstancesintovpc.html>

When you create a VPC, you have the option of specifying its instance tenancy. If you launch an instance into a VPC that has an instance tenancy of dedicated, the instance will always run as a Dedicated Instance on hardware that's dedicated for your use.

For more information about creating a VPC and choosing the tenancy options, see [Create a VPC](https://docs.aws.amazon.com/vpc/latest/userguide/create-vpc.html) in the Amazon VPC User Guide.

You can launch a Dedicated Instance using the Amazon EC2 launch instance wizard.

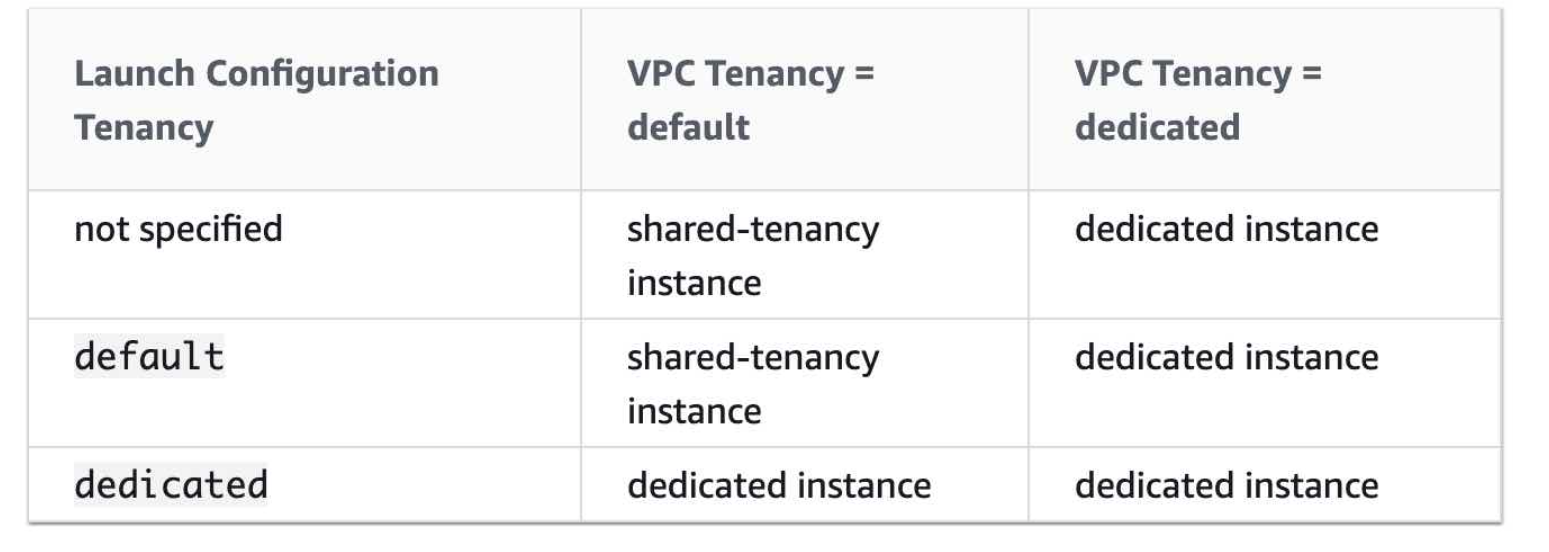


You can change the instance tenancy of a virtual private cloud (VPC) from dedicated to default after you create it.

###### Limitations

* You can't change the instance tenancy of a VPC from default to dedicated after it is created.
* You can't modify the instance tenancy of a VPC using the AWS Management Console You can modify it using the AWS CLI, an AWS SDK, or the Amazon EC2 API.

## Relation on Ec2 launchConfig vs Vpc to set tenancy



# 109 RECORD OF ROUTE53

**Amazon Route 53**

**Amazon Route 53** supports several types of DNS records. Below is a list of all record types available in **Route 53**, along with their usage:

### **1. A (Address) Record**

* **Purpose**: Maps a domain to an **IPv4 address**.
* **Example**:

plaintext

Sao chépChỉnh sửa

example.com → 192.0.2.1

* **Usage**: Used for hosting websites, applications, and services.

### **2. AAAA (IPv6 Address) Record**

* **Purpose**: Maps a domain to an **IPv6 address**.
* **Example**:

plaintext

Sao chépChỉnh sửa

example.com → 2001:db8::ff00:42:8329

* **Usage**: Supports IPv6-enabled networks.

### **3. CNAME (Canonical Name) Record**

* **Purpose**: Maps a **subdomain** to another domain.
* **Example**:

plaintext

Sao chépChỉnh sửa

www.example.com → CNAME → example.net

* **Limitations**:
  + Cannot be used at the **root domain** (apex).
  + Only works for subdomains.

### **4. Alias Record (Amazon-Specific)**

* **Purpose**: Similar to CNAME but can be used for the **root domain**.
* **Example**:

plaintext

Sao chépChỉnh sửa

example.com → Alias → my-load-balancer.amazonaws.com

* **Supported AWS Services**:
  + **Elastic Load Balancer (ELB)**
  + **CloudFront Distributions**
  + **AWS Global Accelerator**
  + **S3 Static Website Hosting**
  + **Route 53 Record Sets**
* **Advantage**: Automatically resolves to an IP address (unlike CNAME).

### **5. MX (Mail Exchange) Record**

* **Purpose**: Directs emails to mail servers.
* **Example**:

plaintext

Sao chépChỉnh sửa

example.com → MX → 10 mail.example.com

* **Usage**: Required for email services like Gmail, Microsoft 365, etc.

### **6. TXT (Text) Record**

* **Purpose**: Stores arbitrary text information.
* **Common Uses**:
  + **SPF (Sender Policy Framework)**
  + **DKIM (DomainKeys Identified Mail)**
  + **DMARC (Email Authentication)**
  + **Verification for Google, AWS, etc.**
* **Example**:

plaintext

Sao chépChỉnh sửa

example.com → TXT → "v=spf1 include:\_spf.google.com ~all"

### **7. PTR (Pointer) Record**

* **Purpose**: Reverse DNS lookup (IP → domain).
* **Example**:

plaintext

Sao chépChỉnh sửa

1.2.0.192.in-addr.arpa → PTR → example.com

* **Usage**: Often used for email servers and security verification.

### **8. NS (Name Server) Record**

* **Purpose**: Defines the authoritative name servers for a domain.
* **Example**:

plaintext

Sao chépChỉnh sửa

example.com → NS → ns-123.awsdns-45.net

* **Usage**: Delegates DNS management to Route 53 or another provider.

### **9. SOA (Start of Authority) Record**

* **Purpose**: Stores domain administration details.
* **Includes**:
  + Primary name server
  + Contact email
  + Serial number (for changes)
  + Refresh, retry, and expiration times
* **Example**:

plaintext

Sao chépChỉnh sửa

example.com → SOA → ns-123.awsdns-45.net admin.example.com 2024032201 7200 900 1209600 86400

### **10. SRV (Service Locator) Record**

* **Purpose**: Specifies services for a domain (e.g., SIP, XMPP, LDAP).
* **Example**:

plaintext

Sao chépChỉnh sửa

\_sip.\_tcp.example.com → SRV → 10 5 5060 sipserver.example.com

* **Usage**: Used in VoIP, messaging, and directory services.

### **11. SPF (Deprecated - Use TXT Instead)**

* **Purpose**: Used for email authentication (now done via TXT).
* **Example**:

plaintext

Sao chépChỉnh sửa

example.com → SPF → "v=spf1 include:\_spf.google.com ~all"

### **12. CAA (Certificate Authority Authorization) Record**

* **Purpose**: Restricts which Certificate Authorities (CAs) can issue SSL certificates.
* **Example**:

plaintext

Sao chépChỉnh sửa

example.com → CAA → 0 issue "letsencrypt.org"

* **Usage**: Prevents unauthorized SSL/TLS certificate issuance.

### **13. DS (Delegation Signer) Record**

* **Purpose**: Used in **DNSSEC** to secure domain delegations.
* **Example**:

plaintext

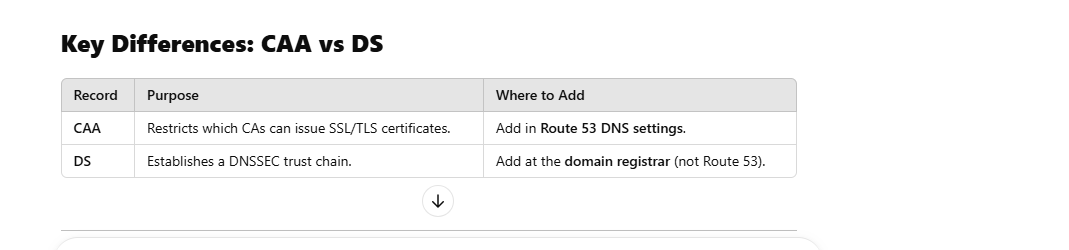
Sao chépChỉnh sửa

example.com → DS → 12345 8 2 abcdef1234567890abcdef1234567890abcdef12

* **Usage**: Validates DNSSEC-signed zones.

## ****Which One Do You Need?****

* **Website Hosting** → A, AAAA, Alias
* **Subdomain Redirection** → CNAME
* **Email Configuration** → MX, TXT (SPF, DKIM, DMARC)
* **Reverse DNS (IP → Domain)** → PTR
* **Security (SSL Restrictions)** → CAA
* **Load Balancer/CDN (AWS)** → Alias



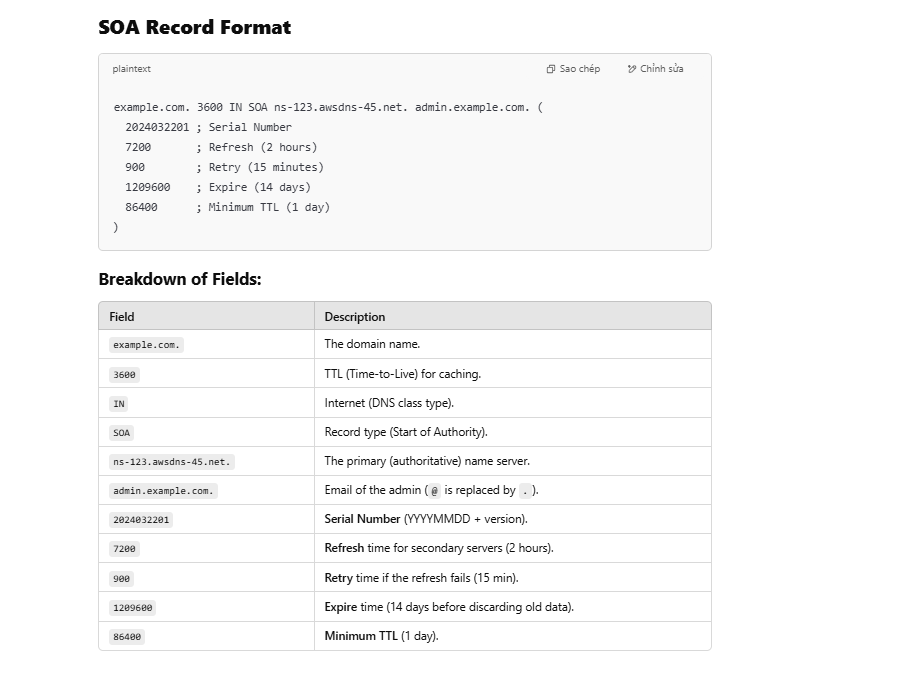
## SOA Detail

**SOA (Start of Authority) record** is always present in a **DNS zone** because it defines the **authoritative information** for that domain. It is required for every DNS domain and is automatically created when you set up a DNS zone in **Route 53** or any other DNS provider.

## ****What Does the SOA Record Do?****

The SOA record contains critical administrative information about the domain, including:

* **Primary Name Server** → The main DNS server for the domain.
* **Admin Email** → Contact email for the domain administrator.
* **Serial Number** → A version number that updates when DNS changes occur.
* **Refresh Time** → How often secondary (slave) servers check for updates.
* **Retry Time** → How long secondary servers wait before retrying a failed update.
* **Expire Time** → How long a secondary server should keep using the old data if it can't reach the primary server.
* **Minimum TTL** → The default caching time for DNS records.



# 110 where kinesis storage

You're right! By default, **Amazon Kinesis Data Streams** retains data for **24 hours**, but it can be extended up to **7 days**. However, AWS later introduced **long-term retention (LTR)**, which allows you to retain data for **up to 365 days**.

Here’s a breakdown of **Kinesis retention options**:

### **1. Kinesis Data Streams Retention Period**

* **Default:** **24 hours**
* **Extended:** **Up to 7 days** (configurable)
* **Long-Term Retention (LTR):** **Up to 365 days** (if enabled)

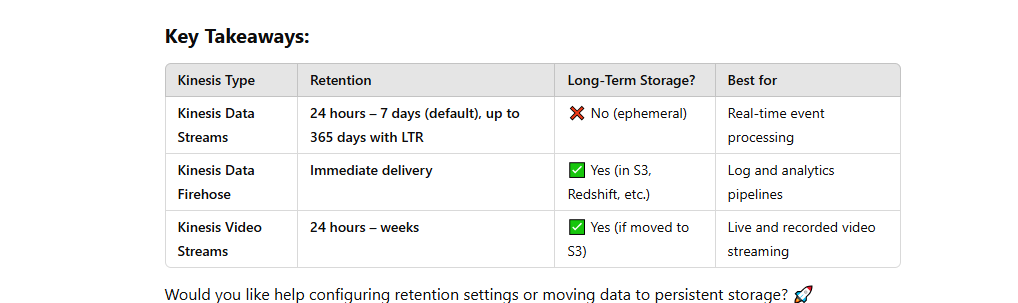
🔹 Data stored in Kinesis Data Streams is not persistent—it gets deleted after the retention period unless explicitly moved to another AWS service like S3.

### **2. Kinesis Data Firehose Retention**

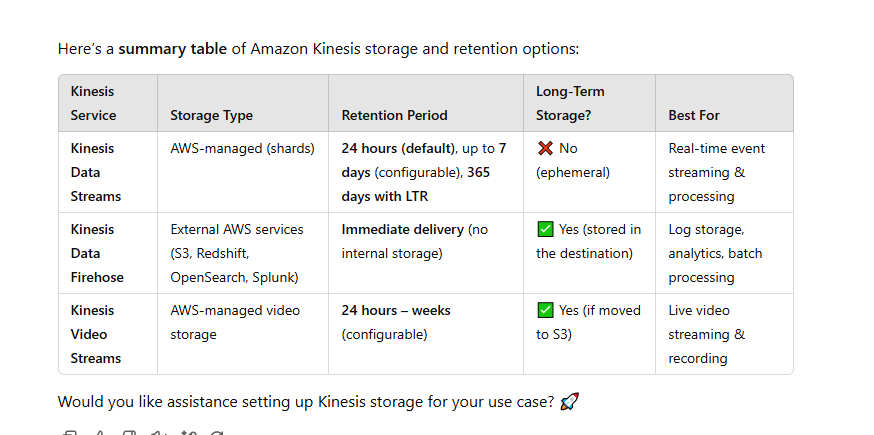
* **Does NOT store data permanently.**
* **Acts as a pipeline** to store data in:
  + Amazon **S3** (Data Lake)
  + Amazon **Redshift** (Data Warehouse)
  + Amazon **OpenSearch Service** (Search & Analytics) ElasticSearch
  + **Splunk** (Monitoring & Logs)
* **Data is stored permanently in the**
* **service**, not in Firehose itself.

### **3. Kinesis Video Streams Retention**

* **Default:** 24 hours
* **Configurable Retention:** Days to weeks
* **For long-term storage**, video streams need to be **moved to Amazon S3**.



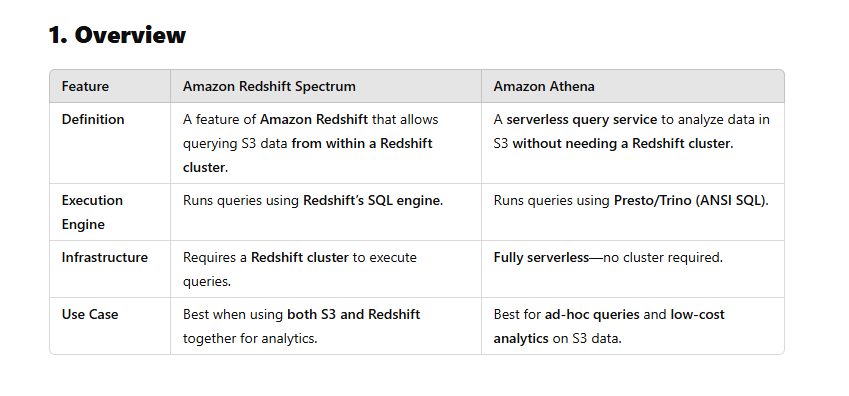
## Summary



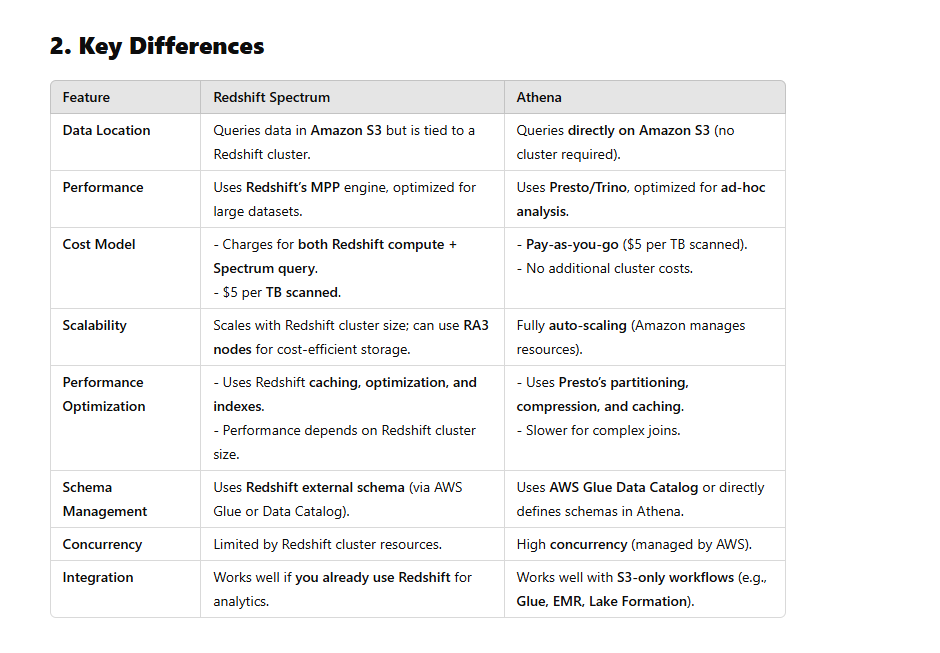
# 111 Diff redshift spectrum and authena

Amazon **Redshift Spectrum** and **Amazon Athena** are both services that allow querying data stored in **Amazon S3** using **SQL**, but they are designed for different use cases. Below is a detailed comparison:

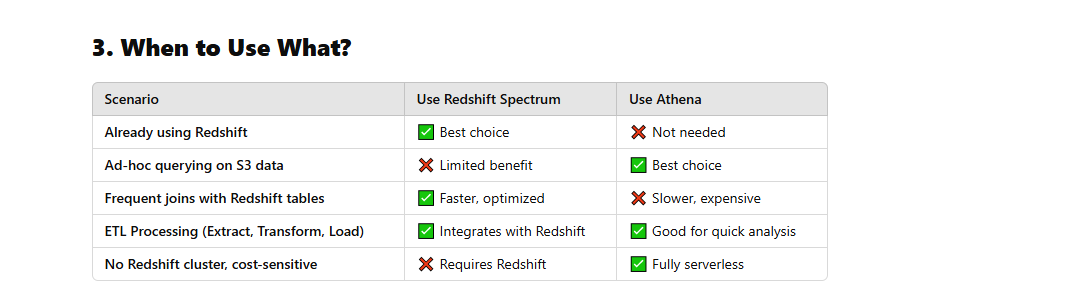
## ****1. Overview****



## Key Differences



## When to Use What?



## ****4. Summary****

* **Use Redshift Spectrum** if:
  + You **already have a Redshift cluster**.
  + You need to **join S3 data with Redshift tables**.
  + You handle **large-scale, frequent analytics**.
* **Use Athena** if:
  + You need a **serverless** solution with **no setup**.
  + You want **ad-hoc querying** on S3.
  + You need to analyze S3 data **without Redshift**.

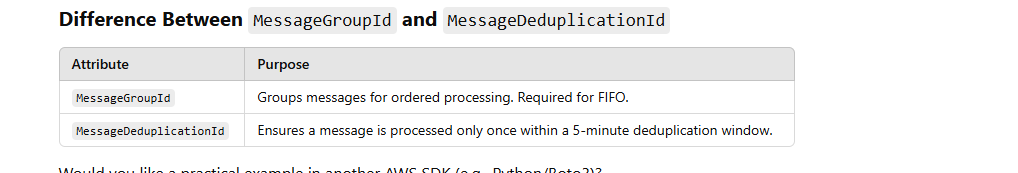
# 112 what is attribute Group ID in SQS in aws

In **AWS SQS (Simple Queue Service)**, the **MessageGroupId** attribute is used in **FIFO (First-In-First-Out) queues** to group related messages together. It ensures that messages within the same group are processed in order and are only delivered to one consumer at a time.

### **Key Points about** MessageGroupId

1. **Ensures Ordered Processing**
   * Messages with the same MessageGroupId are always delivered in the order they are sent.
   * Different groups can be processed in parallel.
2. **Applies Only to FIFO Queues**
   * Standard SQS queues do **not** support this attribute.
   * It is mandatory for FIFO queues.
3. **Sticky Consumer Behavior**
   * Once a consumer starts processing messages from a MessageGroupId, SQS ensures that the same consumer continues processing messages from that group until it finishes.
4. **Example Usage**
   * Suppose you have a banking application where all transactions for a particular **account** must be processed in sequence. You can set MessageGroupId to the **account number** so that messages related to the same account are processed in order.

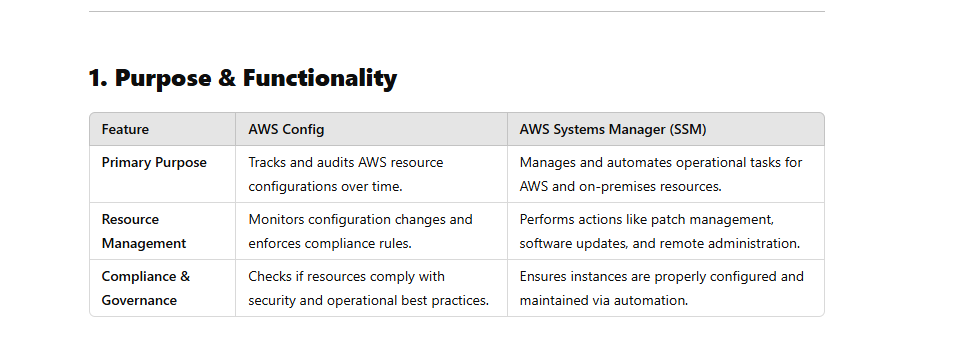
Difference Between MessageGroupId and MessageDeduplicationId



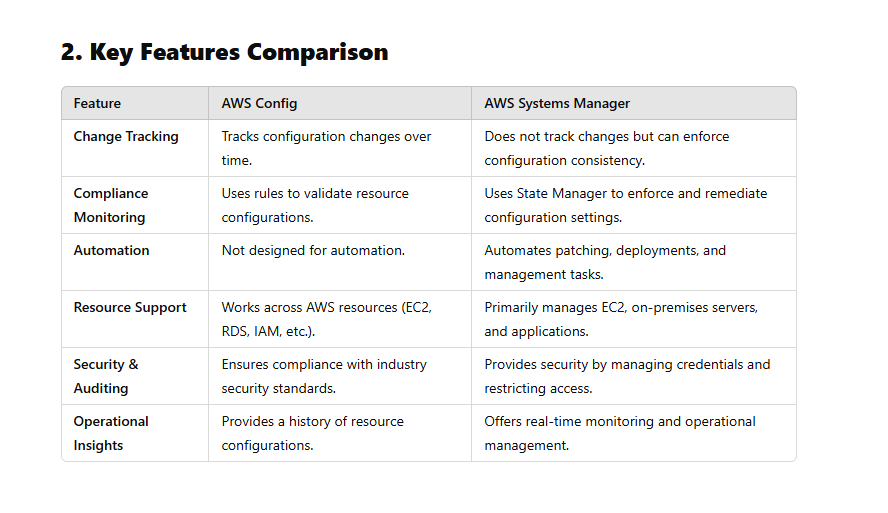
# 113 what diff aws config vs aws system manager

AWS Config and AWS Systems Manager are both AWS services used for resource management and compliance, but they serve different purposes. Here’s a **detailed comparison**:

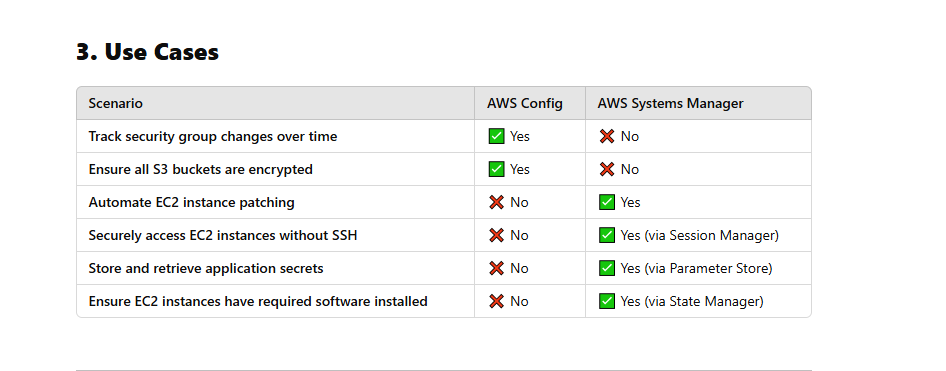
## ****1. Purpose & Functionality****



## Key Features Comparison



## Use Cases



## ****4. When to Use AWS Config vs. AWS Systems Manager****

### **Use AWS Config when:**

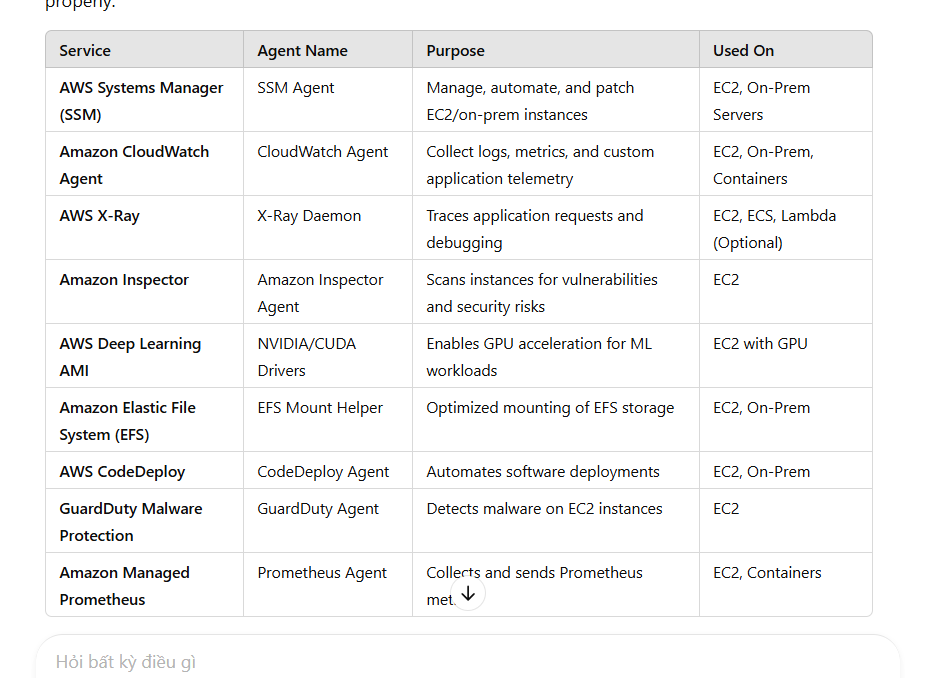
✔️ You need to track changes to AWS resources.  
✔️ You want to audit configurations for compliance (e.g., ensuring encryption is enabled).  
✔️ You need continuous monitoring of resource settings.

### **Use AWS Systems Manager when:**

✔️ You need to manage EC2 instances and on-prem servers.  
✔️ You want to automate patching, software updates, or configurations.  
✔️ You need secure, SSH-free remote access to instances.

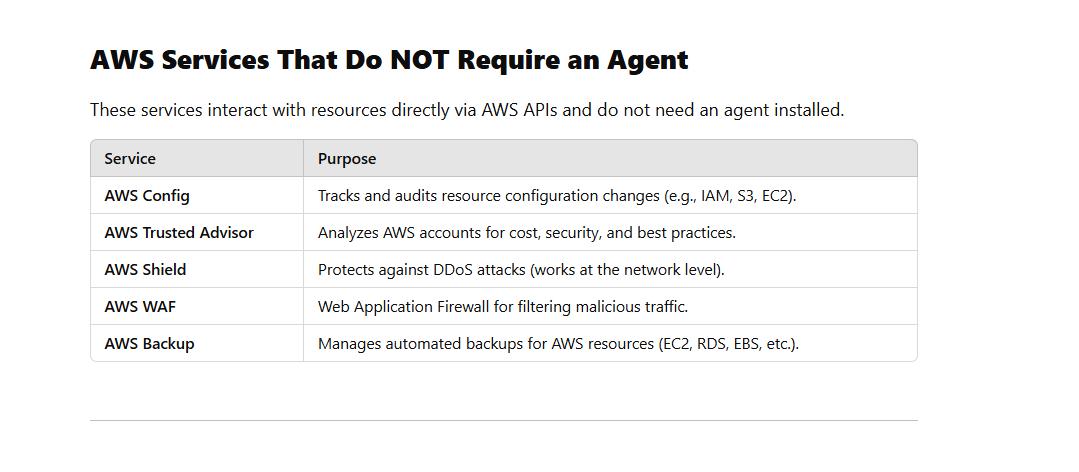
# 114 AWS Services That Require an Agent

These services need an **agent** installed on the resource (EC2, on-prem, or container) to function properly.



## ****AWS Services That Do NOT Require an Agent****

These services interact with resources directly via AWS APIs and do not need an agent installed.



# 115 Diff between AWS workflow and aws step function || AWS pipeline

AWS **Step Functions** and **AWS Workflow Services** (e.g., AWS Managed Workflows for Apache Airflow - MWAA) serve different purposes in orchestrating workflows. Here’s how they differ:

### **AWS Step Functions**

AWS Step Functions is a **serverless** workflow orchestration service that helps in coordinating AWS services using **state machines and tasks**.

🔹 **Best for:**

* Orchestrating AWS services like Lambda, ECS, and DynamoDB.
* Building microservices-based workflows.
* Automating sequential and parallel task execution.

🔹 **Key Features:**

* **State machines** define execution flow (JSON-based Amazon States Language).
* Supports **synchronous & asynchronous** execution.
* **Automatic retries, error handling, and logging** via CloudWatch.
* Integrates with AWS SDK, Lambda, ECS, etc.
* **Pay-per-use pricing** (you pay for the transitions between steps).

🔹 **Example Use Cases:**  
✅ Order processing  
✅ Data processing pipelines  
✅ ETL workflows  
✅ Microservice orchestration

## ****2. Amazon Simple Workflow Service (SWF)**** (Older, More Customizable)

Amazon SWF is a **workflow coordination service** that allows you to run, manage, and track background processes. Unlike Step Functions, SWF gives more **control over execution logic**, but requires you to manage workers.

### 🔹 **Key Features**:

* **Requires custom worker programs** (EC2, on-prem, or AWS Lambda).
* **Decider-based workflow** (separates coordination from execution).
* Supports **human task processing** (e.g., manual approvals).
* **Long-running workflows** (up to 1 year).
* Provides **advanced control over task execution and retries**.

### ✅ **Best for**:

* Workflows needing **human intervention**.
* **Long-running** (stateful) workflows.
* Workflows requiring **custom worker logic**.

## ****3. Amazon EventBridge Pipes**** (Event-Driven Integration)

AWS Pipes is a **point-to-point event routing service** that simplifies connecting event sources to targets **without writing code**.

### 🔹 **Key Features:**

* **Stateless**, **real-time** event processing.
* Connects event sources (SQS, Kinesis, DynamoDB Streams) to targets (Lambda, Step Functions, SQS, etc.).
* **Built-in filtering and enrichment** using AWS Lambda or API Gateway.
* **Low latency** event delivery.
* **Cheaper** than Step Functions (you pay for event processing).

### ✅ **Best for:**

* Simple event-driven integrations.
* Moving data between AWS services in **real-time**.
* **Streaming** workloads (e.g., processing Kinesis events).
* **Low-latency, stateless** workflows.

## ****AWS CodePipeline**** (CI/CD Automation)

AWS CodePipeline is a **continuous integration and continuous delivery (CI/CD) service** used for automating software release processes.

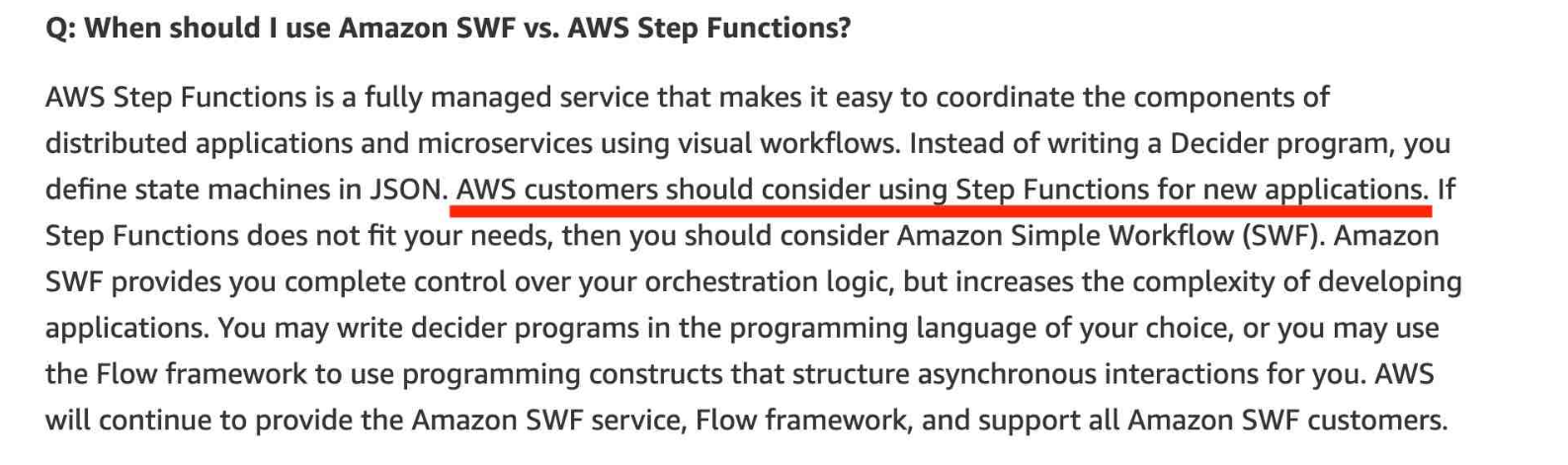
### 🔹 **Key Features**:

* **Used for software deployment workflows** (build → test → deploy).
* **Integrates with AWS CodeBuild, CodeDeploy, GitHub, Jenkins, etc.**
* **Triggers deployments automatically based on code changes**.
* **Supports parallel and sequential actions in a pipeline**.
* **Optimized for DevOps and CI/CD workflows**.

### ✅ **Best for**:

* **Automating software releases** (from code commit to deployment).
* **Managing CI/CD workflows** for applications.
* **Deploying applications to AWS services (ECS, Lambda, EC2, S3, etc.).**

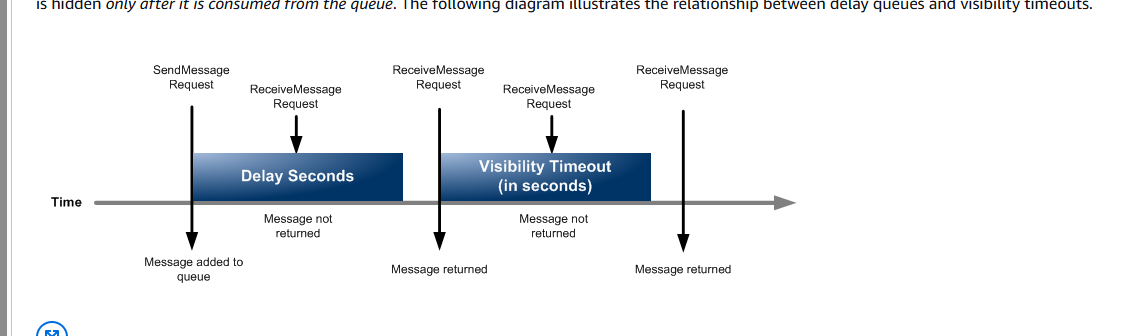
## When should using step function and aws SWF



# 116 SQS Đelay query And Visibility time out

Delay queues let you postpone the delivery of new messages to consumers for a number of seconds, for example, when your consumer application needs additional time to process messages. If you create a delay queue, any messages that you send to the queue remain invisible to consumers for the duration of the delay period

Delay queues are similar to [visibility timeouts](https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html) because both features make messages unavailable to consumers for a specific period of time. The difference between the two is that, for delay queues, a message is hidden when it is first added to queue, whereas for visibility timeouts a message is hidden only after it is consumed from the queue. The following diagram illustrates the relationship between delay queues and visibility timeouts.



Tóm lại delay là trước khi gửi cho consumer

Visibility timeout là đã gửi cho consumer chờ nó báo xử lý xong

# 117 SQS temporary queues

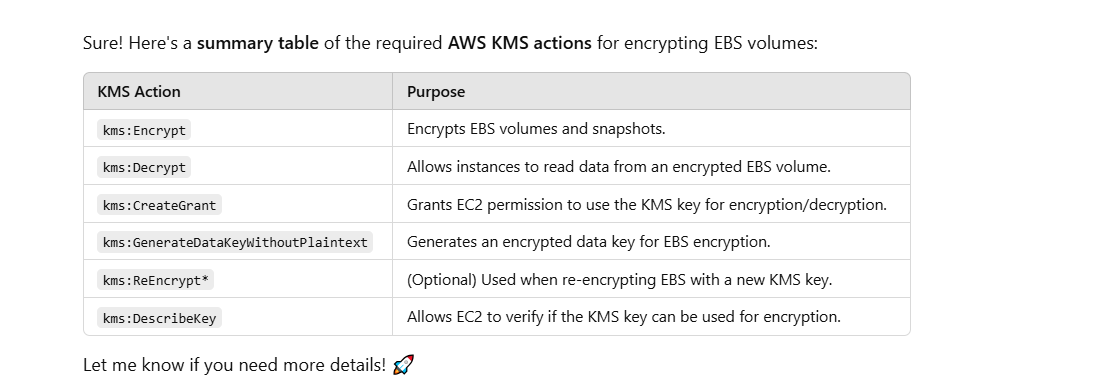
Using temporay queues to more thoughtout when peek time

# 118 EBS Encrypt

<https://docs.aws.amazon.com/ebs/latest/userguide/ebs-encryption-requirements.html>

## Permissions for users

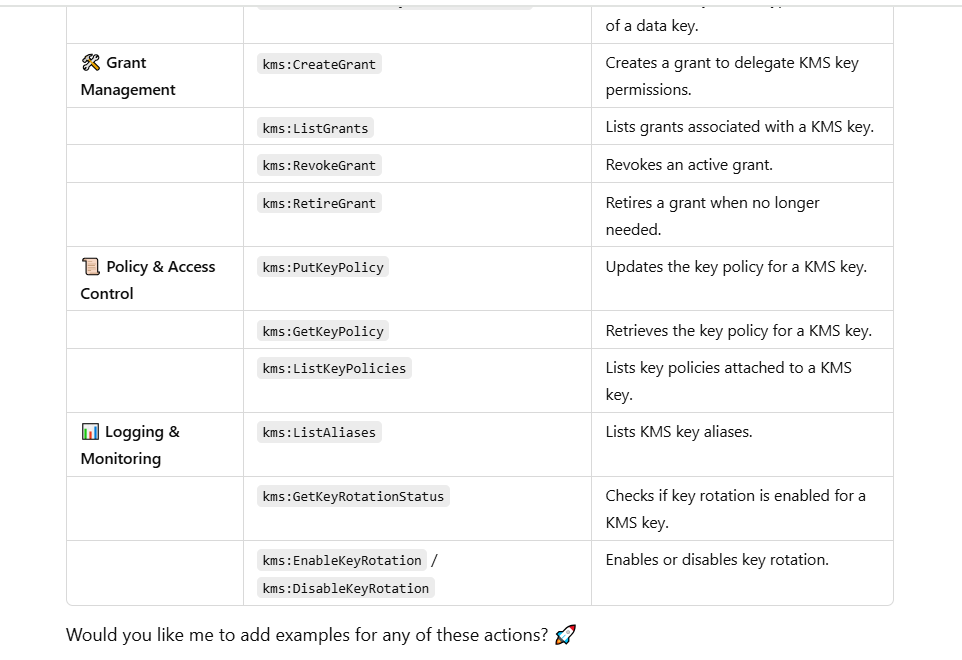
When you use a KMS key for EBS encryption, the KMS key policy allows any user with access to the required AWS KMS actions to use this KMS key to encrypt or decrypt EBS resources. You must grant users permission to call the following actions in order to use EBS encryption:

* kms:CreateGrant
* kms:Decrypt
* kms:DescribeKey
* kms:GenerateDataKeyWithoutPlainText
* kms:ReEncrypt
* 

# 119 All action of KMS

AWS KMS Actions Summary Table





# 120 what is Raid in aws

RAID (Redundant Array of Independent Disks) in AWS refers to the use of RAID configurations on **Amazon Elastic Block Store (EBS) volumes** or **Amazon EC2 instance store volumes** to improve performance, redundancy, or both.

### **Common RAID Levels in AWS**

AWS does not provide built-in RAID support, but you can configure RAID on EC2 instances using Linux or Windows.

1. **RAID 0 (Striping) – Performance Boost**
   * Data is spread across multiple volumes.
   * Increases read/write performance.
   * No redundancy—failure of one volume results in total data loss.
2. **RAID 1 (Mirroring) – High Availability**
   * Data is duplicated across multiple volumes.
   * Provides fault tolerance—if one volume fails, data is still available on the other.
   * No performance gain; requires double the storage capacity.
3. **RAID 5 (Striping with Parity) – Balanced Performance & Fault Tolerance**
   * Uses striping with parity for redundancy.
   * Provides fault tolerance with one disk failure.
   * Not recommended for AWS EBS due to high write penalties.
4. **RAID 6 (Striping with Double Parity) – Extra Fault Tolerance**
   * Can survive two disk failures.
   * Better fault tolerance than RAID 5 but lower write performance.
5. **RAID 10 (Striping + Mirroring) – Best for Performance & Redundancy**
   * Combination of RAID 0 and RAID 1.
   * Provides both performance and fault tolerance.
   * Requires at least four volumes.

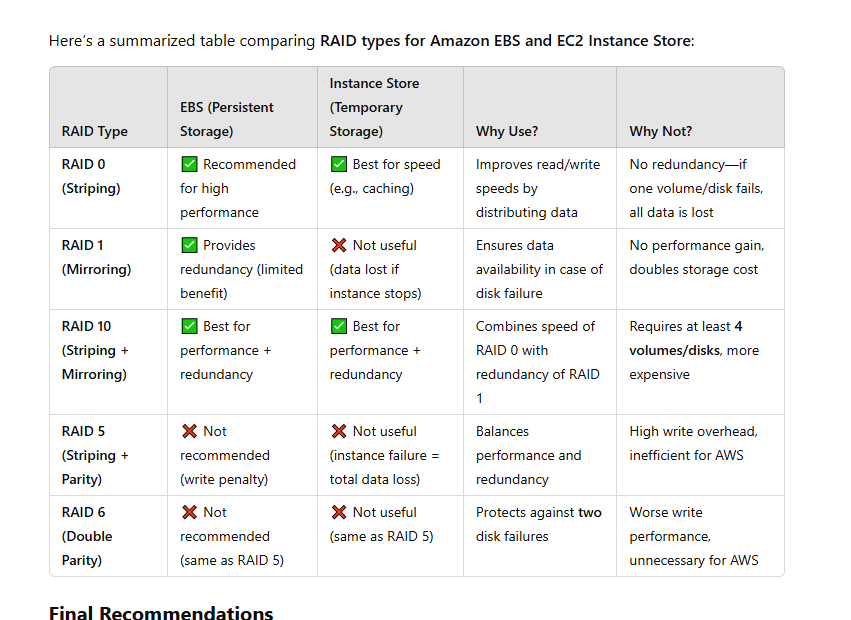
### **When to Use RAID in AWS**

* **RAID 0** for high-speed applications like databases that require high IOPS but can tolerate data loss.
* **RAID 1** for critical data that needs high availability.
* **RAID 10** for applications requiring both performance and redundancy (e.g., transactional databases).

### **Best Practices**

* **Use EBS Snapshots:** AWS recommends using EBS snapshots instead of RAID for data durability.
* **Avoid RAID 5/6 on EBS:** AWS discourages RAID 5/6 because of write performance issues.
* **Monitor Performance:** Use **Amazon CloudWatch** to monitor EBS performance.

**RAID types for Amazon EBS and EC2 Instance Store**:



### **Final Recommendations**

* **For EBS** → Use **RAID 0 for speed** or **RAID 10 for speed + redundancy**. But AWS **recommends using a single large Provisioned IOPS (io2) volume** instead.
* **For Instance Store** → Use **RAID 0** for **maximum speed** or **RAID 10** for **some redundancy**. But remember **data is lost if the instance stops**.

# 121 Using rule actions in AWS WAF

<https://docs.aws.amazon.com/waf/latest/developerguide/waf-rule-action.html>

The rule action tells AWS WAF what to do with a web request when it matches the criteria defined in the rule. You can optionally add custom behavior to each rule action.

Here are the rule action options:

* **Allow** – AWS WAF allows the request to be forwarded to the protected AWS resource for processing and response. This is a terminating action. In rules that you define, you can insert custom headers into the request before forwarding it to the protected resource.
* **Block** – AWS WAF blocks the request. This is a terminating action. By default, your protected AWS resource responds with an HTTP 403 (Forbidden) status code. In rules that you define, you can customize the response. When AWS WAF blocks a request, the Block action settings determine the response that the protected resource sends back to the client.
* **Count** – AWS WAF counts the request but does not determine whether to allow it or block it. This is a non-terminating action. AWS WAF continues processing the remaining rules in the web ACL. In rules that you define, you can insert custom headers into the request and you can add labels that other rules can match against.
* **CAPTCHA and Challenge** – AWS WAF uses CAPTCHA puzzles and silent challenges to verify that the request is not coming from a bot, and AWS WAF uses tokens to track recent successful client responses.

CAPTCHA puzzles and silent challenges can only run when browsers are accessing HTTPS endpoints. Browser clients must be running in secure contexts in order to acquire tokens.

# 122 what is aws system manager

AWS Systems Manager (SSM) is a service that helps you manage and automate operational tasks across your AWS infrastructure. It provides a unified interface for managing AWS resources and on-premises servers, enabling automation, monitoring, and configuration management.

### **Key Features of AWS Systems Manager**

1. **Session Manager** – Provides secure shell (SSH) and Remote Desktop Protocol (RDP) access to EC2 instances without opening inbound ports.
2. **Patch Manager** – Automates patching for operating systems and applications across EC2 and on-prem servers.
3. **Parameter Store** – Securely stores configuration data and secrets.
4. **Automation** – Enables task automation with pre-defined or custom runbooks.
5. **State Manager** – Ensures that your infrastructure remains in a consistent and compliant state.
6. **OpsCenter** – Centralized issue management and operational insights.
7. **Inventory** – Collects metadata from EC2 instances and on-premises servers for compliance and auditing.

### **Use Cases**

* Securely manage EC2 instances without SSH/RDP access.
* Automate software patching and updates.
* Store and retrieve application secrets securely.
* Run scripts and commands on multiple instances at scale.
* Enforce compliance and security best practices.

Would you like details on a specific feature or how to set up AWS Systems Manager for your environment?

# 123 what is Lightsail in aws

Amazon Lightsail is a cloud computing service by AWS that provides an easy-to-use platform for deploying and managing virtual private servers (VPS). It is designed for users who need a simple, cost-effective way to run websites, applications, and development environments without dealing with complex cloud infrastructure.

### Key Features of Lightsail:

* **Pre-configured VPS:** Includes pre-installed OS and applications like WordPress, LAMP, and Node.js.
* **Fixed Pricing:** Offers predictable monthly pricing with different plans.
* **Automatic Backups:** Includes snapshots for data recovery.
* **Networking:** Provides static IPs, DNS management, and load balancers.
* **Scalability:** Can integrate with AWS services like RDS, S3, and CloudFront.
* **Simplified Management:** Uses a user-friendly console with minimal configuration.

### Use Cases:

* Hosting simple websites and blogs.
* Running small business applications.
* Development and testing environments.
* Lightweight web applications and APIs.

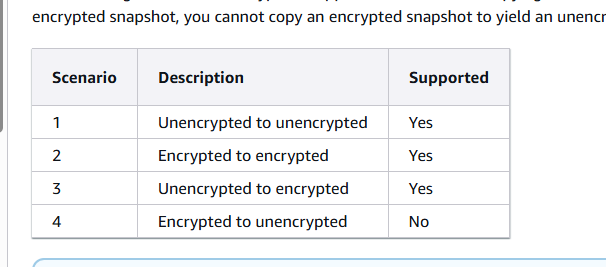
# 124 Copy an Amazon EC2 AMI

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/CopyingAMIs.html>

* You can copy an AMI across AWS Regions ( as long as region in same aws partition)
* You can share an AMI with another AWS account
* Copying an AMI backed by an encrypted snapshot cannot result in an unencrypted target snapshot

## Encryption and copying

The following table shows encryption support for various AMI-copying scenarios. While it is possible to copy an unencrypted snapshot to yield an encrypted snapshot, you cannot copy an encrypted snapshot to yield an unencrypted one.



* , when copying an **instance-store-backed AMI**, you need to specify an **S3 bucket** as an intermediate storage location. This is because **instance-store-backed AMIs** use S3 to store their snapshots, unlike **EBS-backed AMIs**, which use EBS snapshots.

### Steps for Copying an **Instance-Store-Backed AMI**:

1. **Create a new AMI** from the existing instance-store-backed instance.
2. **Upload the AMI bundle** to an S3 bucket.
3. **Register the AMI** in the target region by providing the S3 bucket location.

# 125 what is AWS partition.

In AWS, a **partition** is a top-level grouping of AWS Regions and services that share a common security and compliance boundary. AWS has multiple partitions, and each AWS account belongs to a specific partition.

### **Common AWS Partitions**

1. **aws (Standard AWS Commercial Partition)**
   * This is the default partition for most AWS users.
   * Includes regions like us-east-1, eu-west-1, ap-southeast-1, etc.
2. **aws-cn (AWS China Partition)**
   * Separate from the global AWS commercial partition.
   * Includes regions like cn-north-1 (Beijing) and cn-northwest-1 (Ningxia).
   * Operates under Chinese regulations and requires a separate account.
3. **aws-us-gov (AWS GovCloud Partition)**
   * Designed for U.S. government agencies and contractors.
   * Includes regions like us-gov-west-1 and us-gov-east-1.
   * Meets strict security and compliance requirements.

### **Where is the Partition Used?**

Partitions are used in **Amazon Resource Names (ARNs)** and **IAM policies** to scope resources and permissions.

#### **Example ARN Format**

ruby

Sao chépChỉnh sửa

arn:<partition>:<service>:<region>:<account-id>:<resource>

For different partitions:

* **Commercial AWS:** arn:aws:s3:::my-bucket
* **China AWS:** arn:aws-cn:s3:::my-bucket
* **GovCloud AWS:** arn:aws-us-gov:s3:::my-bucket

### **Why Does Partition Matter?**

* Ensures security and compliance by separating AWS environments.
* Helps in cross-region IAM policies and service configurations.
* Required when writing code or policies that must work across different AWS environments.

# 126 what is Golden AMI

( AMI cài thêm một số thư viện bên ngoài khác sử dụng cho aws beanstreak)

A **Golden AMI (Amazon Machine Image)** refers to a pre-configured, standardized, and optimized AMI that is used as a base for creating new instances on Amazon Web Services (AWS). It contains the essential configurations, software, and security patches that an organization needs.

Golden AMIs are typically created for consistency, security, and automation, ensuring that all instances launched from this image are identical in terms of their configuration. This reduces the risk of human error, improves deployment speed, and provides a streamlined and secure way to scale applications or services in the cloud.

Key features of a Golden AMI include:

1. **Consistency**: Every instance created from the Golden AMI has the same configuration, reducing configuration drift.
2. **Security**: They are configured with the latest security patches and updates.
3. **Efficiency**: They are optimized for faster boot times and better performance.
4. **Custom Software**: They often include custom applications, security agents, or monitoring tools specific to the organization’s needs.

Golden AMIs are used in environments where you need to launch multiple EC2 instances with similar configurations, especially in large-scale, automated, and cloud-native setups.

## Use case

* 3. Faster Instance Boot Time

127 what is Amazon Data Lifecycle Manager

Amazon **Data Lifecycle Manager (DLM)** is an AWS service that helps automate the **creation, retention, and deletion** of **Amazon Elastic Block Store (EBS) snapshots** and **Amazon Machine Images (AMIs)**. It allows you to define **lifecycle policies** to manage backups efficiently, ensuring compliance, cost optimization

# 127 what is diff delicate host vs delicate EC2

It looks like you meant **Dedicated Host vs. Dedicated EC2 Instance** (instead of "delicate" 😃). These terms both refer to **dedicated compute resources in AWS**, but they have key differences.

### **1. Dedicated Host**

A **Dedicated Host** is a **physical server** that is fully allocated to your AWS account. You have complete control over **instance placement**, licensing, and hardware affinity.

#### **Key Features:**

* You get a **full physical machine** (host-level control).
* Allows you to **manually place instances** on the host.
* Helps with **Bring Your Own License (BYOL)** scenarios (Windows, Oracle).
* Ensures **compliance** with strict security and regulatory requirements.
* More expensive but offers **hardware isolation**.

#### **Use Cases:**

* Compliance-heavy industries (e.g., finance, healthcare).
* Workloads needing **full control over physical resources**.
* **BYOL software** that requires dedicated infrastructure.

### **2. Dedicated EC2 Instance**

A **Dedicated EC2 Instance** runs on dedicated hardware but **does not guarantee** that the same physical machine will always be assigned to your instances.

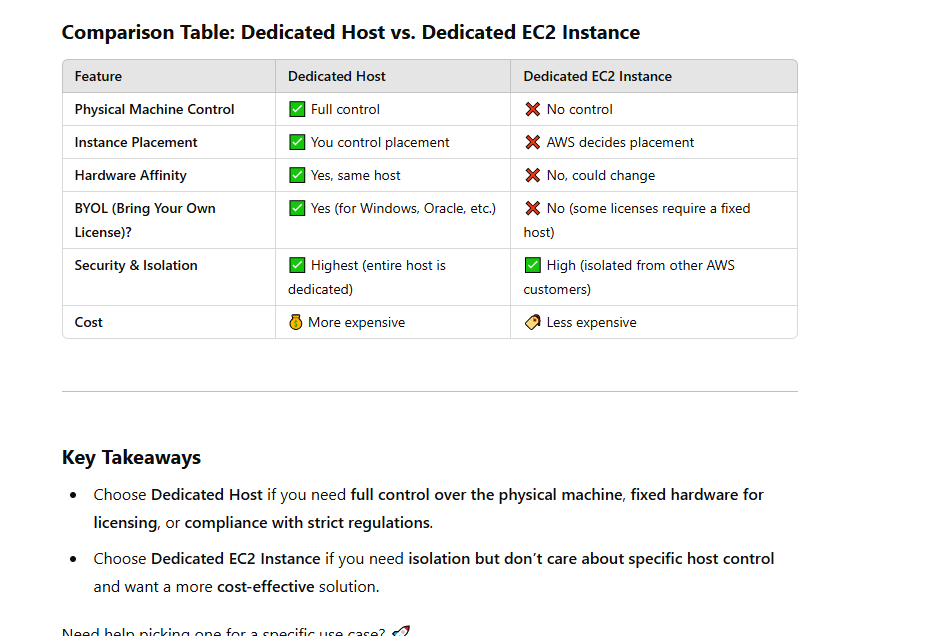
#### **Key Features:**

* Runs on **hardware that is not shared** with other AWS customers.
* AWS manages instance placement; you **don't control** which physical server is used.
* Does **not** support BYOL for some software that requires a **fixed physical host**.
* Cheaper than **Dedicated Hosts** but more expensive than shared EC2.

#### **Use Cases:**

* Workloads needing **isolation from other AWS customers**.
* **Security-sensitive applications** but without the need for host-level control.
* Workloads where **BYOL is not a concern**.

### **Comparison Table: Dedicated Host vs. Dedicated EC2 Instance**



# 128 what is Pilot Light in aws

### **Pilot Light in AWS**

Pilot Light is a disaster recovery (DR) strategy in **AWS** that maintains a **minimal, always-on environment** in the cloud, which can be quickly scaled up in the event of a failure or disaster.

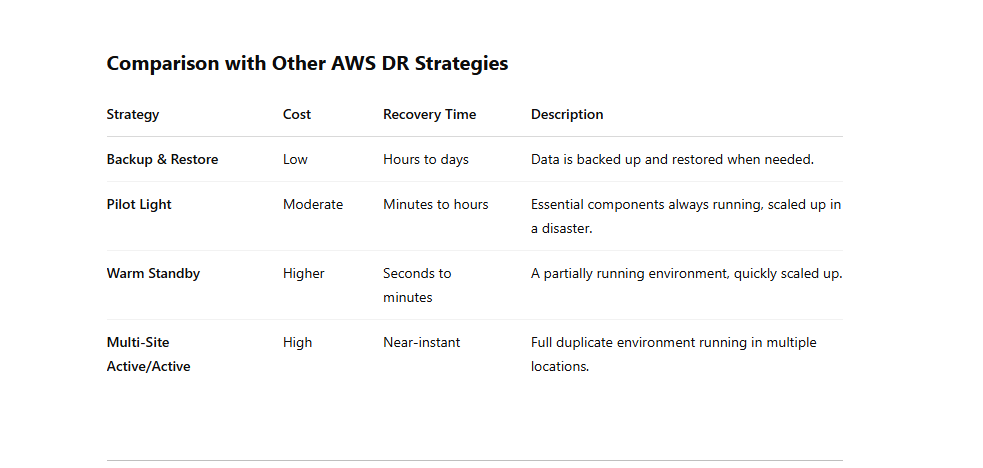
### **How It Works**

* A small **core infrastructure** (like databases and essential services) is kept running in AWS.
* Other **non-essential components** (like application servers) are kept **off** and only started when needed.
* When a disaster occurs, you **scale up** the Pilot Light environment by deploying additional instances to restore full service quickly.

### **Use Case**

* Businesses that need **fast recovery** but don't want the cost of running a full duplicate environment.
* Suitable for applications where **RTO (Recovery Time Objective) is short** but RPO (Recovery Point Objective) can tolerate minor delays.

## Comparison with Other AWS DR Strategies



# 129 AWS Disaster Recovery (DR) Strategies

## ****1. Backup & Restore****

### 🔹 **Features**:

✔️ **Lowest cost** (pay only for storage)  
✔️ **Slowest recovery time** (hours to days)  
✔️ **Data is backed up regularly to S3, Glacier, or another storage service**  
✔️ **Restoration requires deploying infrastructure and applications**

### 🔹 **Use Case**:

* Non-critical applications with low Recovery Time Objective (**RTO**) and Recovery Point Objective (**RPO**)
* Long-term data retention requirements

### 🔹 **AWS Services Used**:

📌 **Amazon S3, Amazon Glacier, AWS Backup, RDS snapshots, DynamoDB backups**

## ****2. Pilot Light****

### 🔹 **Features**:

✔️ **Moderate cost** (some infrastructure always running)  
✔️ **Faster recovery time than Backup & Restore** (minutes to hours)  
✔️ **Critical databases and minimal infrastructure always running**  
✔️ **Application servers are preconfigured but remain turned off**  
✔️ **In case of failure, the environment is quickly scaled up**

### 🔹 **Use Case**:

* Applications requiring **faster recovery** but cost control
* **Critical applications** where database consistency is required

### 🔹 **AWS Services Used**:

📌 **EC2, RDS, DynamoDB, CloudFormation, Auto Scaling, AWS Lambda, EBS Snapshots**

## ****3. Warm Standby****

### 🔹 **Features**:

✔️ **Higher cost than Pilot Light** (some application servers running at reduced capacity)  
✔️ **Faster recovery time** (minutes)  
✔️ **Smaller-scale version of the production environment is always running**  
✔️ **Can scale up rapidly to full production capacity**

### 🔹 **Use Case**:

* Applications that need **quick recovery** without full redundancy costs
* **E-commerce, financial systems**, or services requiring low downtime

### 🔹 **AWS Services Used**:

📌 **EC2 (running at low capacity), RDS Read Replicas, Elastic Load Balancer (ELB), Route 53 for DNS failover, Auto Scaling**

## ****4. Multi-Site (Active-Active)****

### 🔹 **Features**:

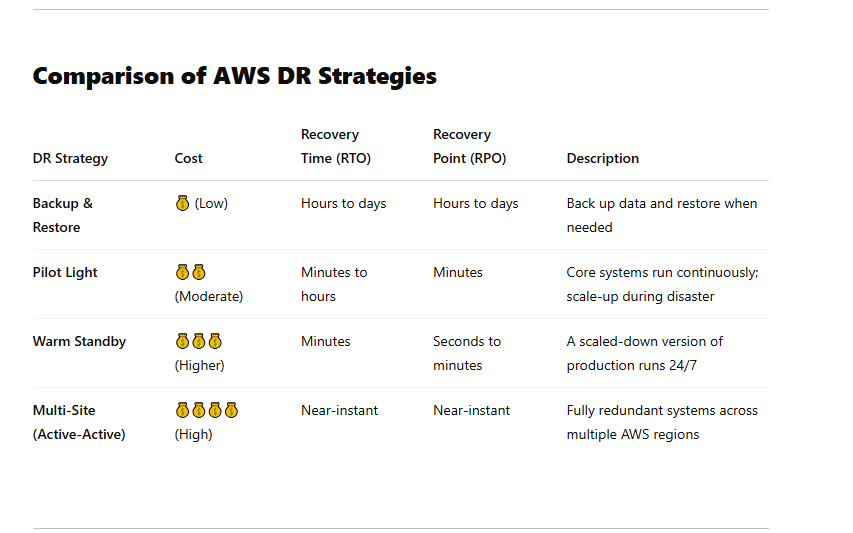
✔️ **Highest cost** (fully replicated environment in multiple AWS regions)  
✔️ **Fastest recovery time** (near-zero RTO)  
✔️ **Traffic is actively distributed across multiple AWS regions**  
✔️ **No downtime even in case of failure**

### 🔹 **Use Case**:

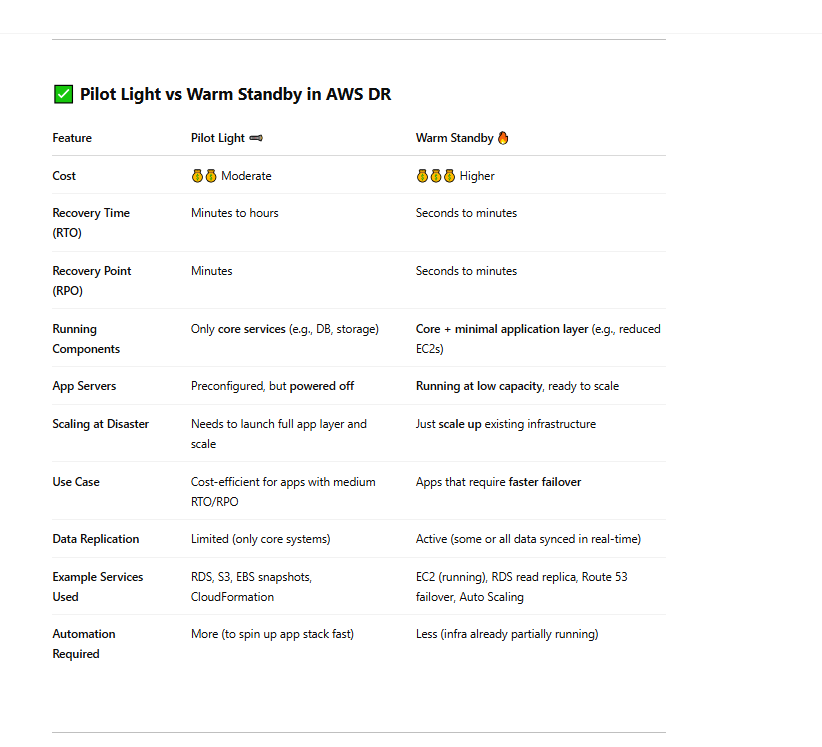
* **Mission-critical applications** with **zero downtime** tolerance
* **Global applications** requiring redundancy in multiple AWS regions

### 🔹 **AWS Services Used**:

📌 **AWS Global Accelerator, Route 53, Elastic Load Balancer, DynamoDB Global Tables, Aurora Global Database, Auto Scaling across regions**



## Pilot Light vs Warm Standby in AWS DR



### 🧠 Quick Analogy:

* **Pilot Light** = Like having a stove with just the flame on—ready to cook but not cooking yet.
* **Warm Standby** = Like cooking on low heat—it's already running but needs to crank up to full.

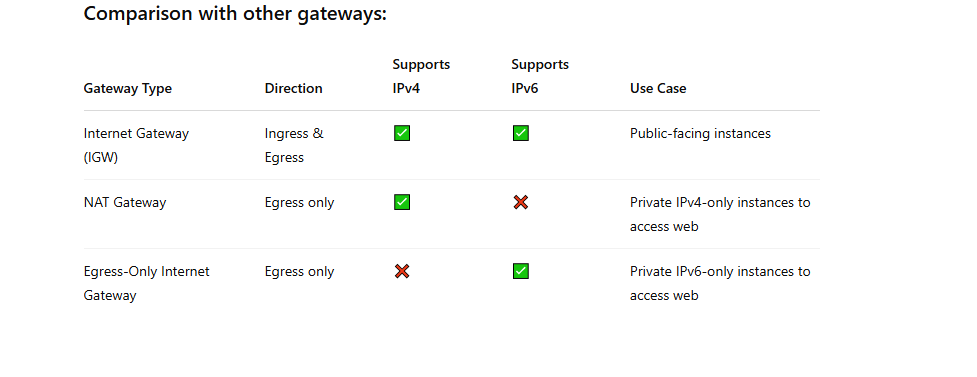
# 130 what is Egress-Only Internet Gateways

An **Egress-Only Internet Gateway (EOIGW)** is a **VPC component in AWS** that allows instances in a **VPC with IPv6 addresses** to **access the internet for outbound traffic**, but **prevents the internet from initiating connections back** to those instances (i.e., **no inbound access**).

### Key points:

* 🛡️ **Outbound only**: It allows only **egress (outbound)** IPv6 traffic to the internet.
* 🚫 **No ingress**: It does **not allow incoming** connections from the internet (unlike a regular Internet Gateway).
* 🌐 **IPv6 only**: It works **only for IPv6** traffic, not IPv4.
* 🔒 **Use case**: Ideal for instances that need to fetch updates, reach external services, etc., but **should not be directly reachable** from the public internet.

### Comparison with other gateways:



# 131 **Summary of When to Use Aurora vs. DynamoDB**:

### **Use Cases for Amazon Aurora**:

Amazon Aurora is a **relational database** that is compatible with MySQL and PostgreSQL. It's best suited for applications that require strong consistency, relational data models, and complex queries. Aurora supports **Auto-scaling** with Aurora Serverless, making it ideal for variable or unpredictable workloads.

#### **1. Traditional Web Applications**:

* **Use Case**: eCommerce platforms, content management systems (CMS), and customer relationship management (CRM) systems.
* **Why Aurora**: These types of applications often require transactional consistency, complex queries, and relationships between entities (e.g., users, orders, products). Aurora’s compatibility with MySQL and PostgreSQL makes it easy to migrate from on-premise databases to the cloud while benefiting from high availability, durability, and automatic scaling.

#### **2. Business Intelligence & Analytics**:

* **Use Case**: Data warehousing, reporting, and dashboards.
* **Why Aurora**: Aurora supports complex SQL queries and can handle large volumes of structured data, making it ideal for analytic workloads. It provides excellent read scalability with **Aurora Replicas** and high availability for critical analytics applications.

#### **3. SaaS Applications**:

* **Use Case**: Multi-tenant SaaS platforms that need scalable relational data storage.
* **Why Aurora**: With its ability to automatically scale compute and storage and offer low-latency reads and writes, Aurora works well for SaaS applications with unpredictable traffic patterns. The ability to isolate tenants using schema-based partitioning or a multi-tenant database model is another key benefit.

#### **4. High Availability and Disaster Recovery**:

* **Use Case**: Applications that need high availability and quick recovery times.
* **Why Aurora**: Aurora’s built-in support for **Multi-AZ** deployments ensures high availability and automatic failover, making it ideal for applications that need 24/7 uptime. It also automatically handles storage replication and backup.

#### **5. Auto-Scaling Web Apps**:

* **Use Case**: Web applications with unpredictable or variable traffic, such as media platforms, gaming backends, or seasonal promotions.
* **Why Aurora**: Aurora Serverless enables auto-scaling based on usage, ensuring your application can handle sudden spikes in traffic without requiring manual intervention. It automatically adjusts to match the demand, providing cost efficiency during low-traffic periods.

### **Use Cases for Amazon DynamoDB**:

Amazon DynamoDB is a **NoSQL database** designed for applications that require high throughput, low-latency responses, and scalability without complex relational data models. DynamoDB shines in scenarios with massive scale and where data access patterns are simple (key-value or document-based).

#### **1. Mobile Applications**:

* **Use Case**: User profile storage, leaderboards, real-time data synchronization.
* **Why DynamoDB**: DynamoDB provides seamless scaling for mobile applications, particularly those with millions of users or rapid read/write operations. It also supports high throughput and low-latency access to data, which is crucial for mobile experiences (e.g., chat apps, gaming).

#### **2. IoT (Internet of Things) Applications**:

* **Use Case**: Storing sensor data, telemetry, and device logs.
* **Why DynamoDB**: IoT applications often involve high volumes of data with varying traffic patterns. DynamoDB can handle the massive scale of writes (e.g., sensor data) and scale automatically to accommodate spikes in data ingestion, making it ideal for real-time IoT data storage.

#### **3. Real-Time Analytics**:

* **Use Case**: Real-time data processing, clickstream data, and online recommendation systems.
* **Why DynamoDB**: DynamoDB's high availability and low-latency performance make it a great choice for applications that need to handle a large amount of real-time data and perform rapid analysis on that data. DynamoDB Streams can be used to trigger real-time processing workflows with AWS Lambda or other systems.

#### **4. Session Management & Caching**:

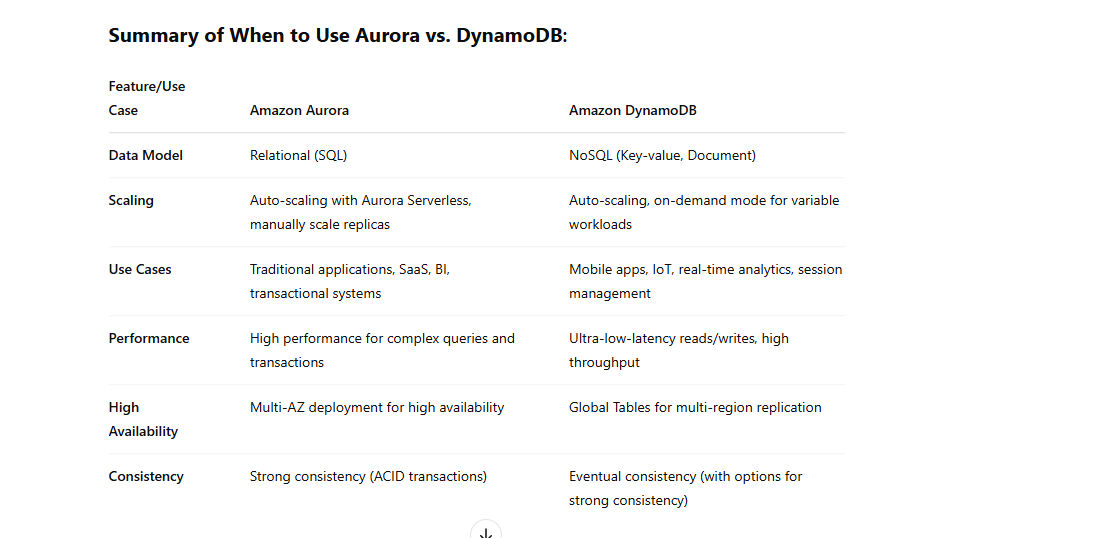
* **Use Case**: Storing user sessions, shopping cart contents, or caching frequently accessed data.
* **Why DynamoDB**: DynamoDB is often used for session management due to its fast read and write capabilities and ability to scale. It’s particularly useful for applications that need to store transient data with low-latency access.

#### **5. Serverless Applications**:

* **Use Case**: Backend for serverless architectures, such as Lambda-based workflows.
* **Why DynamoDB**: DynamoDB works seamlessly with AWS Lambda in serverless applications, where you don’t need to manage the database infrastructure. DynamoDB’s **on-demand** capacity mode ensures that the database can automatically scale to match traffic spikes without provisioning.

#### **6. Global Applications**:

* **Use Case**: Applications that require global reach and low-latency access to data across regions.
* **Why DynamoDB**: DynamoDB offers **Global Tables**, allowing data to be replicated across AWS regions for low-latency reads and writes. This makes it ideal for globally distributed applications, ensuring consistent performance across regions.



### Conclusion:

* **Amazon Aurora** is best for applications that need relational databases, complex queries, transactions, and scalability. It’s suited for businesses running traditional applications, analytics, and systems requiring strong consistency and high availability.
* **Amazon DynamoDB** is perfect for applications that require fast access to large amounts of unstructured or semi-structured data, with high scalability and low latency. It works well for real-time applications, serverless architectures, and applications with large-scale distributed data needs.

# 131 How can i authenticate lamda

To authenticate an **AWS Lambda function** to access **other services**, you have several options depending on whether the service is:

* Another **AWS service** (e.g., S3, DynamoDB, SNS, etc.)
* An **external service** (e.g., third-party API, your own REST/gRPC microservice)

Here's a comprehensive list of **all ways you can authenticate Lambda to another service**, grouped by type:

## ✅ ****1. AWS Service Authentication Methods****

### **a. IAM Role (Lambda Execution Role)**

* **Best for**: Accessing AWS services (S3, DynamoDB, SQS, etc.)
* **How it works**: Lambda assumes an IAM role with attached policies.
* **Setup**:
  + Go to Lambda > Configuration > Permissions
  + Assign a role with policies (e.g., AmazonS3ReadOnlyAccess)
  + SDKs use this automatically (no need to manually sign requests)

### **b. IAM AssumeRole (Cross-account access)**

* **Best for**: Accessing services in **another AWS account**
* **How it works**: Lambda assumes a role in another account via sts:AssumeRole
* **Setup**:
  + Role in target account trusts Lambda's IAM role
  + Use AWS STS in Lambda to assume the role

### **c. AWS SigV4 Signing**

* **Best for**: Manually signing HTTP requests to AWS services
* **How it works**: You use AWS SDK or manually generate SigV4 signed headers.
* **Use case**: Custom HTTP clients, calling services like Elasticsearch, API Gateway, etc.

### **d. Resource-based Policies**

* **Best for**: Granting Lambda permissions to invoke another service like:
  + S3 bucket triggers
  + SNS topic subscriptions
  + API Gateway invocation
* **How it works**: The target service has a policy that allows the Lambda’s IAM role to access it.

## ✅ ****2. AWS to AWS Private Authentication****

### **e. VPC and Security Group-Based Access**

* **Best for**: Private communication with RDS, Redshift, or services behind ALB/NLB
* **Setup**:
  + Place Lambda in a VPC
  + Control access via security groups

## ✅ ****3. External Services Authentication Methods****

### **f. API Key**

* **How it works**: Lambda adds API key in request header
* **Use case**: Third-party REST APIs, internal microservices

### **g. OAuth 2.0 / JWT**

* **How it works**: Lambda includes Bearer token (access token) in request
* **Use case**:
  + Call Google APIs, Microsoft Graph, etc.
  + Internal microservices using OAuth2/JWT

### **h. Basic Auth**

* **How it works**: Lambda uses base64-encoded username:password in header
* **Use case**: Internal APIs, services with legacy auth

### **i. mTLS (Mutual TLS)**

* **How it works**: Lambda authenticates using a client certificate
* **Setup**:
  + Place Lambda behind API Gateway with custom domain + mTLS
  + Call external services that require TLS client auth

### **j. Signed JWT (Custom Auth)**

* **How it works**: Lambda generates and signs JWT with private key
* **Use case**:
  + Communicate with services expecting signed tokens (e.g., Firebase, custom auth system)

### **k. Amazon Cognito**

* **How it works**: Lambda uses access token from Cognito User Pools or Identity Pools
* **Use case**:
  + Authenticate to APIs protected by Cognito

### **l. Custom Header/Token**

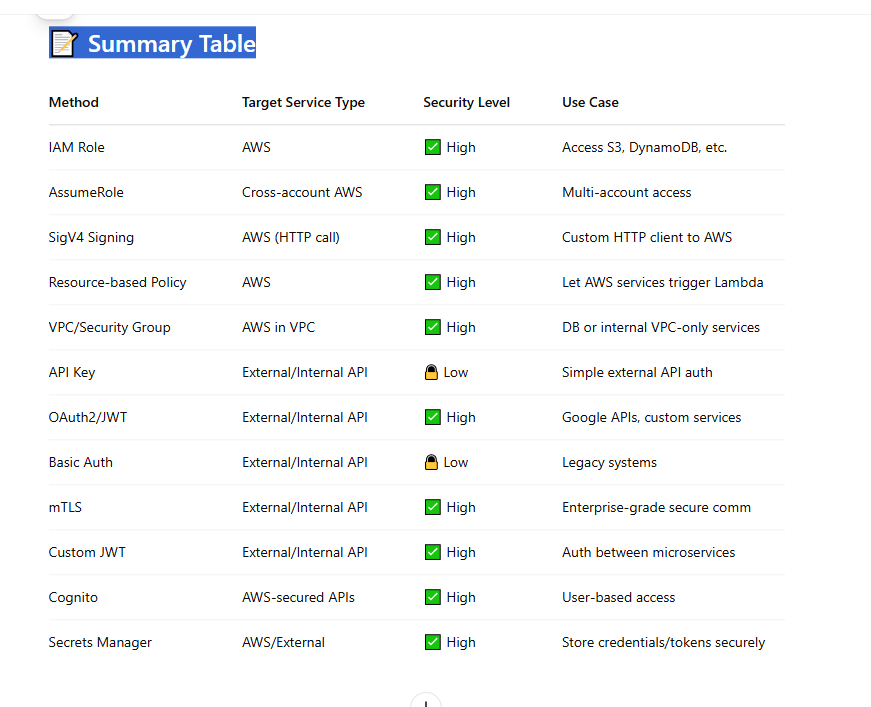
* **How it works**: Lambda injects a known secret/token in request headers
* **Use case**: Internal service-to-service calls

## ✅ ****4. Other Mechanisms****

### **m. Secrets Manager / Parameter Store**

* **How it helps**: Securely store credentials or tokens used in Lambda
* **Use case**: Retrieve DB passwords, API keys, or OAuth tokens securely

📝 Summary Table



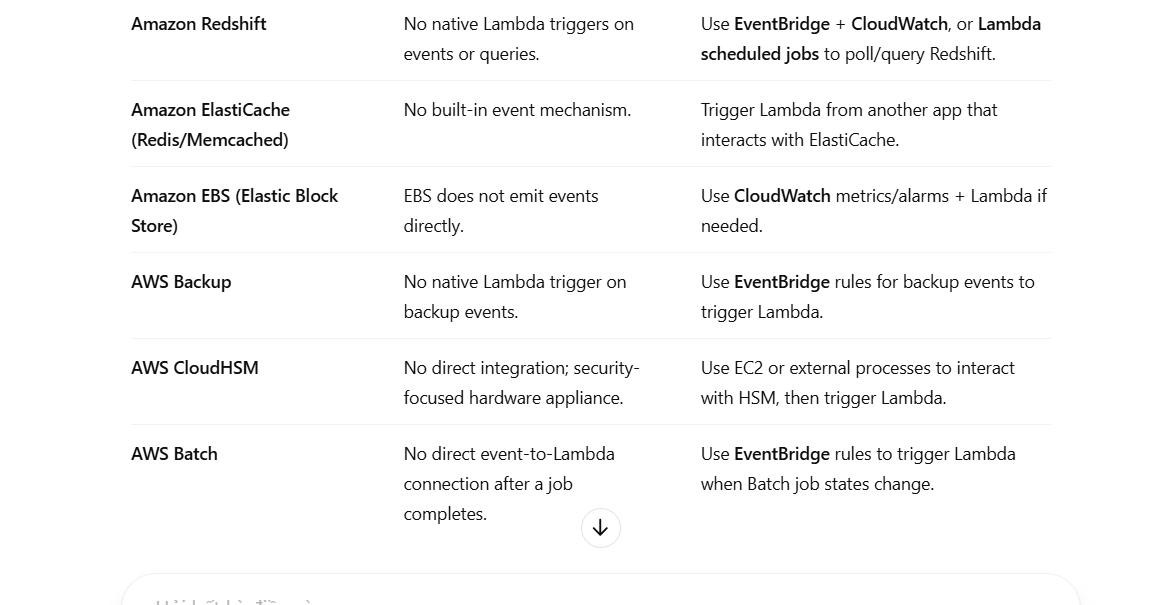
# 132 API GATEWAY throttling request

# <https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-throttling.html>

# 133 AWS services that do not directly support Lambda

🚫 Services That Don’t Natively Trigger Lambda





# 134 when using eventbridge to emit to lamda

When custom application and other service in aws not support lamda : ex ( ses, rds, .. using eventbride instead)

# 135 Types of access control in S3:



# 136 Types of AWS Cognito

## 🔹 Types of AWS Cognito

Amazon Cognito has **two main components**:

### 1. **Cognito User Pools ( For Authen)**

* It's a **user directory** that helps manage **sign-up and sign-in** functionalities.
* Supports authentication through:
  + Email/phone and password
  + Social identity providers (e.g., Google, Facebook, Apple)
  + SAML-based enterprise identity providers
* Includes built-in security features like:
  + Multi-factor authentication (MFA)
  + Account recovery
  + Password policies

### 2. **Cognito Identity Pools** (Federated Identities) (for author)

* Used for **authorization** to grant temporary **AWS credentials** to users so they can access AWS services.
* Supports:
  + Users from Cognito User Pools
  + Social identity providers
  + SAML providers
  + Unauthenticated (guest) users
* Integrates with AWS IAM roles for fine-grained access control.

## 🔹 Key Features

# 137 \*ELB health check

The engineering team at a logistics company is working on a shipments application deployed on a fleet of Amazon EC2 instances behind an Auto Scaling Group (ASG). While configuring new changes for an upcoming release, a team member has noticed that the ASG is not terminating an unhealthy instance.

As a Solutions Architect, which of the following options would you suggest to troubleshoot the issue? (Select three)

* **The health check grace period for the instance has not expired**

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/healthcheck.html#health-check-grace-period>

* **The instance has failed the ELB health check status**
* **The instance maybe in Impaired status**

**The instance maybe in Impaired status** - Amazon EC2 Auto Scaling does not immediately terminate instances with an Impaired status. Instead, Amazon EC2 Auto Scaling waits a few minutes for the instance to recover. Amazon EC2 Auto Scaling might also delay or not terminate instances that fail to report data for status checks. This usually happens when there is insufficient data for the status check metrics in Amazon CloudWatch.

**The instance has failed the ELB health check status** - By default, Amazon EC2 Auto Scaling doesn't use the results of ELB health checks to determine an instance's health status when the group's health check configuration is set to EC2. As a result, Amazon EC2 Auto Scaling doesn't terminate instances that fail ELB health checks. If an instance's status is OutofService on the ELB console, but the instance's status is Healthy on the Amazon EC2 Auto Scaling console, confirm that the health check type is set to ELB.

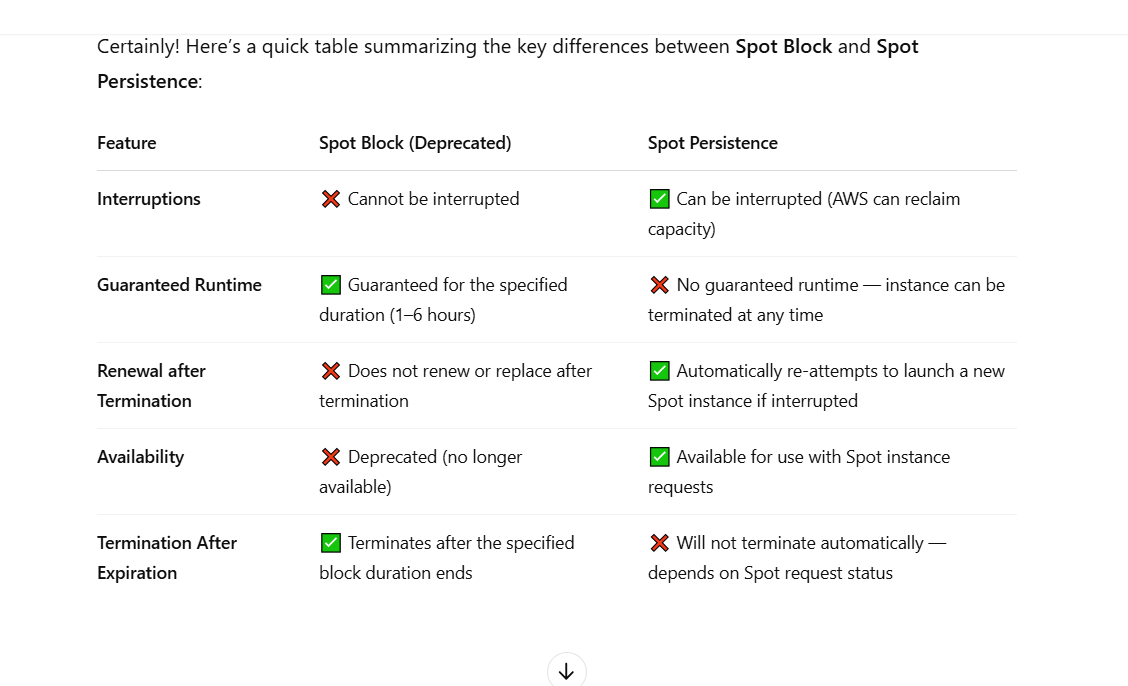
| **Health Status** | **Meaning** |
| --- | --- |
| InService / healthy | Target is healthy and receiving traffic |
| OutOfService / unhealthy | Target is failing health checks and not receiving traffic |
| unused | Target is not in use (not part of any routing config) |
| initial | Target is registering or still in first-time health check phase |
| draining | Target is deregistering; connections are being drained |

# 138 Spot EC2 Request

-

|  |
| --- |
|  |

|  |
| --- |
| **Spot Block (Deprecated)** |



* Nội dung tóm tắt như sau

+ Spot block ( hết hạn ko dùng nữa) ko thể interupt trong thời gian request (1- 6 h gì đó)

+ Spot persistance, spot instance ( nếu vẫn còn yêu cầu Request EC2 instance nó sẽ tự động renew)

 **Persistent Spot Request**:

* If you create a **persistent Spot request**, AWS will **keep trying** to launch a new instance (of the same type or different types, depending on your settings) whenever the instance is interrupted or terminated.
* If your instance is interrupted or terminated (either by AWS or manually), **AWS will re-launch it** without you having to submit a new request.

 **Spot Instance Interruption**:

* When the Spot instance is interrupted by AWS, it **will not be automatically terminated** (unless you do it manually).
* But as long as the **Spot request remains active** (without cancellation), AWS will **attempt to launch another Spot instance** to fulfill the request, based on availability.

# 139 User Data (EC2)

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html>

**By default, scripts entered as user data are executed with root user privileges** - Scripts entered as user data are executed as the root user, hence do not need the sudo command in the script. Any files you create will be owned by root; if you need non-root users to have file access, you should modify the permissions accordingly in the script.

**By default, user data runs only during the boot cycle when you first launch an instance** - By default, user data scripts and cloud-init directives run only during the boot cycle when you first launch an instance. You can update your configuration to ensure that your user data scripts and cloud-init directives run every time you restart your instance.

### Update the instance user data

To update the instance user data, you must first stop the instance. If the instance is running, you can view the user data but you cannot modify it.

###### Warning

When you stop an instance, the data on any instance store volumes is erased. To keep data from instance store volumes, be sure to back it up to persistent storage.

# 140 Blue/green deployment

References:

<https://aws.amazon.com/blogs/networking-and-content-delivery/using-aws-global-accelerator-to-achieve-blue-green-deployments>

<https://docs.aws.amazon.com/codedeploy/latest/userguide/deployments.html>

<https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-policy.html#routing-policy-weighted>

Blue/green deployment is a technique for releasing applications by shifting traffic between two identical environments running different versions of the application: "Blue" is the currently running version and "green" the new version. This type of deployment allows you to test features in the green environment without impacting the currently running version of your application. When you’re satisfied that the green version is working properly, you can gradually reroute the traffic from the old blue environment to the new green environment. Blue/green deployments can mitigate common risks associated with deploying software, such as downtime and rollback capability.

**Use AWS Global Accelerator to distribute a portion of traffic to a particular deployment** - AWS Global Accelerator is a network layer service that directs traffic to optimal endpoints over the AWS global network, this improves the availability and performance of your internet applications. It provides two static anycast IP addresses that act as a fixed entry point to your application endpoints in a single or multiple AWS Regions, such as your Application Load Balancers, Network Load Balancers, Elastic IP addresses or Amazon EC2 instances, in a single or in multiple AWS regions.

AWS Global Accelerator uses endpoint weights to determine the proportion of traffic that is directed to endpoints in an endpoint group, and traffic dials to control the percentage of traffic that is directed to an endpoint group (an AWS region where your application is deployed).

While relying on the DNS service is a great option for blue/green deployments, it may not fit use-cases that require a fast and controlled transition of the traffic. Some client devices and internet resolvers cache DNS answers for long periods; this DNS feature improves the efficiency of the DNS service as it reduces the DNS traffic across the Internet, and serves as a resiliency technique by preventing authoritative name-server overloads. The downside of this in blue/green deployments is that you don’t know how long it will take before all of your users receive updated IP addresses when you update a record, change your routing preference or when there is an application failure.

With AWS Global Accelerator, you can shift traffic gradually or all at once between the blue and the green environment and vice-versa without being subject to DNS caching on client devices and internet resolvers, traffic dials and endpoint weights changes are effective within seconds.

# 141 AWS Cost Explorer Resource Optimization

**AWS Cost Explorer Resource Optimization** is a feature within AWS Cost Explorer that helps you identify cost-saving opportunities by analyzing your usage patterns and recommending **resource right-sizing**—especially for **Amazon EC2 instances**.

### 🔍 What It Does:

* **Analyzes your EC2 usage** over time (default is 14 days).
* Identifies underutilized instances based on CPU, memory, and networking metrics.
* **Recommends downsizing**, stopping, or changing instance types to reduce costs.
* Suggests **Savings Plans** or **Reserved Instances** if applicable.

### 📌 Key Benefits:

* **Cost Savings**: Helps avoid overpaying for unused capacity.
* **Actionable Insights**: Gives you clear recommendations with potential savings.
* **Customizable Filters**: You can filter by instance family, region, or account.
* **Linked Accounts Support**: Useful for organizations with multiple AWS accounts.

### 💡 Use Case Example:

If you have an EC2 instance that is consistently using only 5% of its CPU, Cost Explorer Resource Optimization might recommend:

“You can downgrade this m5.2xlarge instance to m5.large and save 75% monthly.”

### 🚀 How to Access:

1. Go to [AWS Cost Explorer](https://console.aws.amazon.com/cost-management/home).
2. Navigate to **"Resource Optimization"** under the **"Rightsizing recommendations"** section.

Let me know if you want help interpreting a recommendation or automating cost optimization with tools like **AWS Compute Optimizer** or **Lambda scripts**.

# 142 AWS Compute Optimizer

**AWS Compute Optimizer** is a service provided by Amazon Web Services that helps you **optimize your cloud resources** for performance and cost efficiency. It uses machine learning to analyze your **usage patterns** and recommends changes to your AWS compute resources such as:

* **EC2 instances**
* **Auto Scaling groups**
* **EBS volumes**
* **Lambda functions**
* **Amazon ECS services (on Fargate)**

### 🧠 How it works

It monitors your usage over time (usually 14+ days of data) and provides recommendations like:

* **Rightsizing**: Suggests smaller or larger instance types.
* **Changing instance families**: Like moving from general purpose to compute-optimized.
* **Upgrading to newer generation** instances for better performance/cost.
* **EBS volume adjustments** based on throughput and IOPS needs.
* **Lambda memory/timeout tweaks** for better cost/performance balance.

### ✅ Benefits

* **Save money** by eliminating underutilized resources.
* **Improve performance** by fixing bottlenecks or overcommitted resources.
* **Get actionable insights** without manually digging through CloudWatch or billing data.

### 🔒 Security

It doesn’t change anything automatically—it just gives **recommendations**. You apply the changes manually (or automate it via scripts if you want).

### Example Use Case

If you’re running a t3.xlarge EC2 instance that’s only using 20% CPU consistently, Compute Optimizer might suggest switching to a t3.medium to save money.

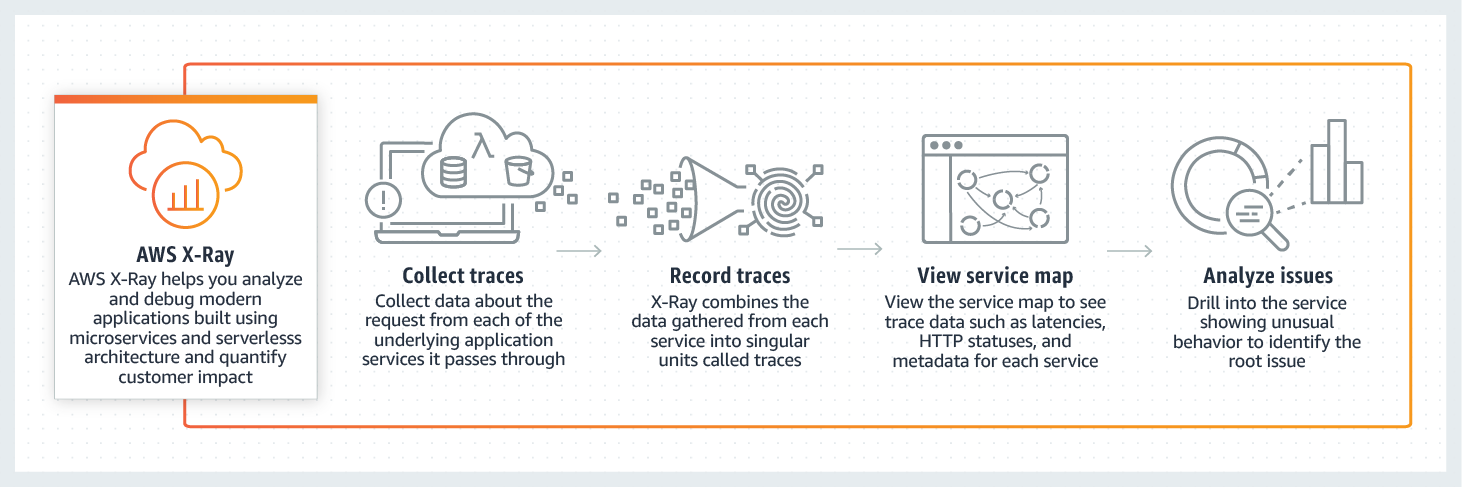
Want to know how to enable or use it for a specific service like EC2 or Lambda?

4o

# 143 **X-Ray (same Lamda)**

AWS X-Ray helps developers analyze and debug production, distributed applications, such as those built using a microservices architecture. With X-Ray, you can understand how your application and its underlying services are performing to identify and troubleshoot the root cause of performance issues and errors. X-Ray provides an end-to-end view of requests as they travel through your application, and shows a map of your application’s underlying components.

You can use X-Ray to collect data across AWS Accounts. The X-Ray agent can assume a role to publish data into an account different from the one in which it is running. This enables you to publish data from various components of your application into a central account.

How X-Ray Works:  via - <https://aws.amazon.com/xray/>

# 144 **Caching strategies**

## 🧠 1. ****Prewarm the Cache****

### ✅ Purpose:

Avoid "cold start" latency by loading expected data into the cache **before** real user traffic hits.

### 🛠️ How to Implement:

* **Manual**: At app startup, load commonly accessed data.
* **Scheduled Job**: Use a cron/scheduler to regularly fetch and cache data.

java

Sao chépChỉnh sửa

// Example in Java + Spring

@PostConstruct

public void prewarmCache() {

List<Product> products = productService.getTopProducts();

products.forEach(product -> cacheService.put(product.getId(), product));

}

## 🚀 2. ****Lazy Loading (Cache on Access)****

### ✅ Purpose:

Only cache data **when it's requested**, then store it for next time.

### 🛠️ How to Implement:

js

Sao chépChỉnh sửa

// Example in Node.js (simple)

function getUser(userId) {

if (cache.has(userId)) return cache.get(userId);

const user = db.getUser(userId);

cache.set(userId, user);

return user;

}

### ⚠️ Downside:

First request is still slow (cold start).

## 🧹 3. ****Cache Expiration / TTL (Time-To-Live)****

### ✅ Purpose:

Automatically remove stale data from cache after a set time.

### 🛠️ How:

* Most cache systems (Redis, Memcached) support TTL per key.
* Example in Redis CLI:

sh

Sao chépChỉnh sửa

SET user:123 "John" EX 3600 # Expires in 1 hour

## 🔄 4. ****Write-Through Cache****

### ✅ Purpose:

Update cache **immediately** when the database is written to.

### 🛠️ How to Implement:

java

Sao chépChỉnh sửa

public void updateUser(User user) {

db.save(user);

cache.put(user.getId(), user); // sync cache

}

### 💡 Benefit:

Cache always has fresh data.

## 🔁 5. ****Read-Through Cache****

### ✅ Purpose:

The cache **handles loading** if data is missing.

### 🛠️ How:

With Redis or a library like **Caffeine (Java)** or **Guava**:

java

Sao chépChỉnh sửa

LoadingCache<String, User> cache = CacheBuilder.newBuilder()

.expireAfterWrite(10, TimeUnit.MINUTES)

.build(userId -> db.getUser(userId));

## 🧯 6. ****Cache Invalidation (Eviction)****

### ✅ Purpose:

Remove or refresh cache entries when underlying data changes.

### 🛠️ Example:

java

Sao chépChỉnh sửa

public void deleteUser(Long id) {

db.delete(id);

cache.evict(id);

}

### Strategies:

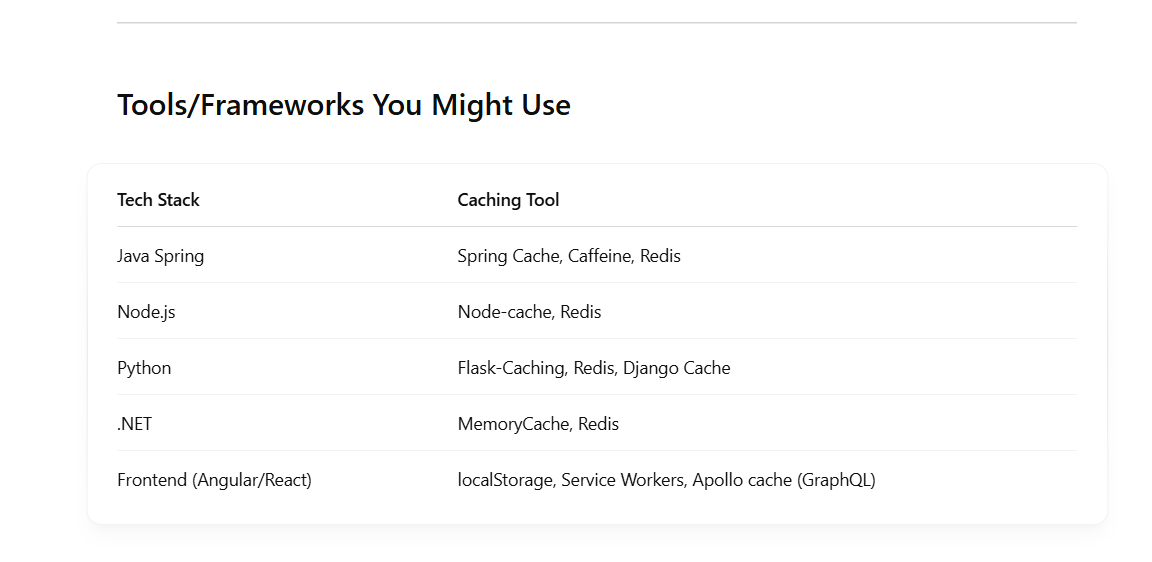
* Manual eviction
* Pattern-based deletion (user:\*)
* Pub/Sub cache invalidation (for distributed systems)

## 🧰 7. ****Hybrid Approach****

Combine strategies:

* Prewarm top N products
* Lazy-load everything else
* TTL for automatic cleanup
* Write-through to keep cache fresh

## Tools/Frameworks You Might Use



# 145 CloudWatch recovery

Absolutely! **Amazon CloudWatch EC2 Recovery** is a feature that allows **automatic recovery of an EC2 instance** if it becomes **impaired** due to underlying hardware or system issues.

### ✅ What is **EC2 Instance Recovery** via CloudWatch?

It’s a **CloudWatch action** that automatically **recovers an EC2 instance** when a **system status check** fails. This is especially helpful when the instance becomes unreachable due to hardware or networking issues **but the EBS volume is intact**.

### 🔧 What it Does:

* **Stops and restarts the instance** on new healthy hardware.
* **Preserves instance metadata**:
  + Instance ID stays the same
  + IP address (Elastic IP/static private IP) remains the same
  + EBS root and attached volumes persist
* **Downtime is minimized** (recovery typically takes a few minutes).

### 💡 Use Case Example:

You have a critical EC2 instance and want to make sure it recovers automatically from hardware-level issues without manual intervention.

### 🚨 Important:

This only recovers the instance when the **system status check fails**, not if the **instance status check** fails (e.g., OS crash or app-level issues).

### 🔄 How to Set Up CloudWatch Recovery

You set up a **CloudWatch Alarm** with an **EC2 recovery action**. Here's how:

#### 🔹 1. Create an Alarm (via Console or CLI)

* **Metric**: StatusCheckFailed\_System
* **Threshold**: >= 1 for 1 consecutive period
* **Action**: EC2 Recovery

#### 🔹 2. Example (AWS CLI):

bash

Sao chépChỉnh sửa

aws cloudwatch put-metric-alarm \

--alarm-name "RecoverMyInstance" \

--metric-name StatusCheckFailed\_System \

--namespace AWS/EC2 \

--statistic Minimum \

--period 60 \

--threshold 1 \

--comparison-operator GreaterThanOrEqualToThreshold \

--dimensions Name=InstanceId,Value=i-1234567890abcdef0 \

--evaluation-periods 1 \

--alarm-actions arn:aws:automate:region:ec2:recover \

--unit Count

### ✅ Supported Instance Types

* Must be an **EBS-backed** instance.
* Some older generation instance types may not support recovery.
* Instance must be in the **"running"** state and **not part of an Auto Scaling Group** (ASG handles recovery differently).

# 146 Case using RDS replica

1. Scaling beyond the compute or I/O capacity of a single DB instance for read-heavy database workloads. This excess read traffic can be directed to one or more read replicas.
2. Serving read traffic while the source DB instance is unavailable. If your source DB Instance cannot take I/O requests (e.g. due to I/O suspension for backups or scheduled maintenance), you can direct read traffic to your read replica(s). For this use case, keep in mind that the data on the read replica may be “stale” since the source DB Instance is unavailable.
3. Business reporting or data warehousing scenarios; you may want business reporting queries to run against a read replica, rather than your primary, production DB Instance.
4. You may use a read replica for disaster recovery of the source DB instance, either in the same AWS Region or in another Region.

# 147 Status Of Replica RDS

The status of a read replica can be one of the following:

* **replicating** – The read replica is replicating successfully.
* **replication degraded (SQL Server and PostgreSQL only) –**Replicas are receiving data from the primary instance, but one or more databases might be not getting updates. This can occur, for example, when a replica is in the process of setting up newly created databases. It can also occur when unsupported DDL or large object changes are made in the blue environment of a blue/green deployment.

The status doesn't transition from replication degraded to error, unless an error occurs during the degraded state.

* **error** – An error has occurred with the replication. Check the **Replication Error** field in the Amazon RDS console or the event log to determine the exact error. For more information about troubleshooting a replication error, see [Troubleshooting a MySQL read replica problem](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.Troubleshooting.html).
* **terminated (MariaDB, MySQL, or PostgreSQL only)** – Replication is terminated. This occurs if replication is stopped for more than 30 consecutive days, either manually or due to a replication error. In this case, Amazon RDS terminates replication between the primary DB instance and all read replicas. Amazon RDS does this to prevent increased storage requirements on the source DB instance and long failover times.

Broken replication can affect storage because the logs can grow in size and number due to the high volume of errors messages being written to the log. Broken replication can also affect failure recovery due to the time Amazon RDS requires to maintain and process the large number of logs during recovery.

* **terminated (Oracle only)** – Replication is terminated. This occurs if replication is stopped for more than 8 hours because there isn't enough storage remaining on the read replica. In this case, Amazon RDS terminates replication between the primary DB instance and the affected read replica. This status is a terminal state, and the read replica must be re-created.
* **stopped (MariaDB or MySQL only)** – Replication has stopped because of a customer-initiated request.
* **replication stop point set (MySQL only)** – A customer-initiated stop point was set using the [mysql.rds\_start\_replication\_until](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/mysql-stored-proc-replicating.html#mysql_rds_start_replication_until) stored procedure and the replication is in progress.
* **replication stop point reached (MySQL only)** – A customer-initiated stop point was set using the [mysql.rds\_start\_replication\_until](https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/mysql-stored-proc-replicating.html#mysql_rds_start_replication_until) stored procedure and replication is stopped because the stop point was reached.

You can see where a DB instance is being replicated and if so, check its replication status. On the **Databases** page in the RDS console, it shows **Primary** in the **Role** column. Choose its DB instance name. On its detail page, on the **Connectivity & security** tab, its replication status is under **Replication**.

# 148 what is Amazon Quicksight

Amazon QuickSight is a **cloud-based business intelligence (BI) service** offered by AWS (Amazon Web Services). It enables you to easily create and publish **interactive dashboards**, visualizations, and reports from your data. Here's a breakdown of what it offers:

### 🔍 Key Features:

* **Data Visualization**: Create charts, graphs, and dashboards with a drag-and-drop interface.
* **Scalable**: Automatically scales to handle thousands of users without managing infrastructure.
* **ML Insights**: Built-in machine learning features like anomaly detection and forecasting.
* **SPICE Engine**: Super-fast, Parallel, In-memory Calculation Engine for quick performance even on large datasets.
* **Embedded Analytics**: Dashboards can be embedded into apps and portals.
* **Multi-Source Support**: Connects to various data sources including:
  + AWS services (S3, RDS, Redshift, Athena)
  + External databases (MySQL, PostgreSQL, etc.)
  + SaaS apps (Salesforce, Excel files, etc.)

### ✅ Common Use Cases:

* Real-time dashboards for operations and KPIs
* Reports for business stakeholders
* Customer-facing analytics within your application

### 🛠️ How It’s Used:

1. **Connect Data**: Link to data sources via direct query or SPICE.
2. **Build Visuals**: Use a web interface to design dashboards.
3. **Share**: Publish dashboards and control access with IAM or user groups.

# 149 IAM database authentication for MariaDB, MySQL, and PostgreSQL

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.IAMDBAuth.html>

You can authenticate to your DB instance using AWS Identity and Access Management (IAM) database authentication. With this authentication method, you don't need to use a password when you connect to a DB instance. Instead, you use an authentication token. An authentication token is a unique string of characters that Amazon RDS generates on request. Each token has a lifetime of 15 minutes. You don't need to store user credentials in the database, because authentication is managed externally using IAM.

# 150 AWS Firewall Manager

AWS Firewall Manager is a security management service that makes it easier to centrally configure and manage firewall rules across multiple AWS accounts and resources. It's particularly useful for organizations that manage a large number of accounts and need to ensure consistent firewall settings and policies across all of them.

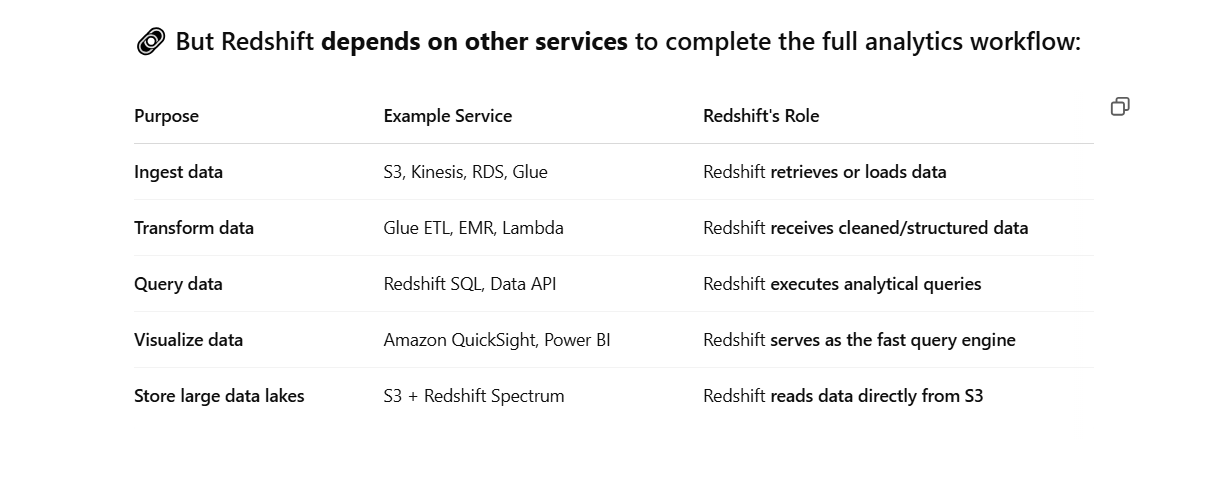
Here are the types of firewall rules and key features of AWS Firewall Manager:

### Types of Firewall Rules:

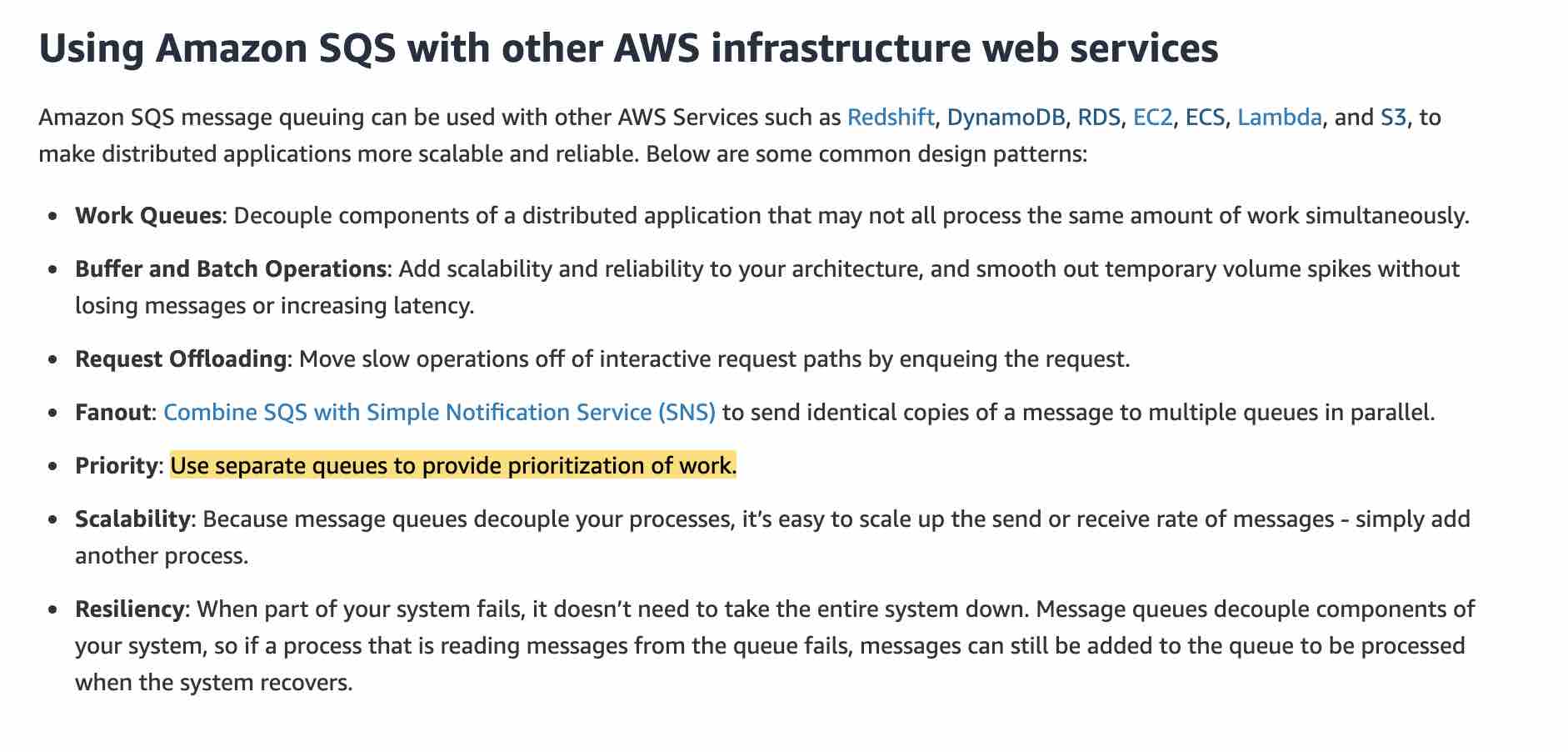
1. **AWS WAF (Web Application Firewall) Rules:**
   * Allows you to configure and manage web application firewall rules across multiple accounts to protect applications from common web exploits (e.g., SQL injection, cross-site scripting).
   * You can apply rules across your entire AWS Organization to manage traffic filtering consistently.
2. **AWS Shield Advanced:**
   * Manages AWS Shield Advanced protection across accounts.
   * Provides DDoS protection with detailed attack diagnostics and automatic mitigation.
3. **VPC Security Group Rules:**
   * Lets you enforce security group rules across multiple accounts.
   * Security groups act as virtual firewalls to control inbound and outbound traffic to AWS resources such as EC2 instances.
4. **AWS Network Firewall:**
   * Lets you manage network firewall policies and protections across multiple accounts within your AWS Organization.
   * Network firewall offers stateful inspection and rule-based filtering for network traffic across VPCs.

# 151 Redshift **depends on other services** to complete the full analytics workflow:

-Note ý này: Redshift ko dùng để phân tích dữ liệu mà nó là dataware house phục vụ cho việc phân tích dữ liệu, các data được đẩy vào từ các service có thể là



# 152 Using SQS in usecase



# 153 EFS Thougput and performance specifycation

## 🔧 ****EFS Performance Modes****

EFS has two performance modes:

### 1. **General Purpose (default)**

* Best for latency-sensitive use cases (e.g., web servers, CMS, home directories).
* **Low latency** for file operations.
* Designed for single-digit millisecond latencies.

### 2. **Max I/O**

* Best for high-throughput workloads with many concurrent accesses (e.g., big data, media processing).
* **Higher latency** than General Purpose.
* Scales to higher levels of aggregate throughput and IOPS.

You select this at the time of file system creation and **cannot change** it afterward.

## 🚀 ****Throughput Modes****

EFS supports two throughput modes:

### 1. **Bursting Throughput (default)**

* Throughput scales with the size of your file system.
* Up to **100 MB/s per TiB** of storage.
* Every file system earns burst credits; can burst up to **100 MB/s** for short periods even with small storage sizes.

### 2. **Provisioned Throughput**

* You can set a specific throughput (independent of storage size).
* Ideal for workloads that require consistent high throughput even with a small amount of data stored.
* Up to **1 GiB/s** throughput currently supported.

## 📈 ****IOPS (Input/Output Operations Per Second)****

* Scales with throughput.
* Amazon states **tens of thousands of IOPS** supported.
* **Baseline and burst IOPS** are driven by throughput (1 MiB/s throughput ≈ 256 IOPS for 4 KiB I/O size).
* EFS is optimized for **large read and write operations** (e.g., 128 KiB).

## 🕒 ****Latency****

* For General Purpose mode: typically **single-digit milliseconds**.
* For Max I/O mode: **higher and more variable latency** due to higher parallelization and network load.

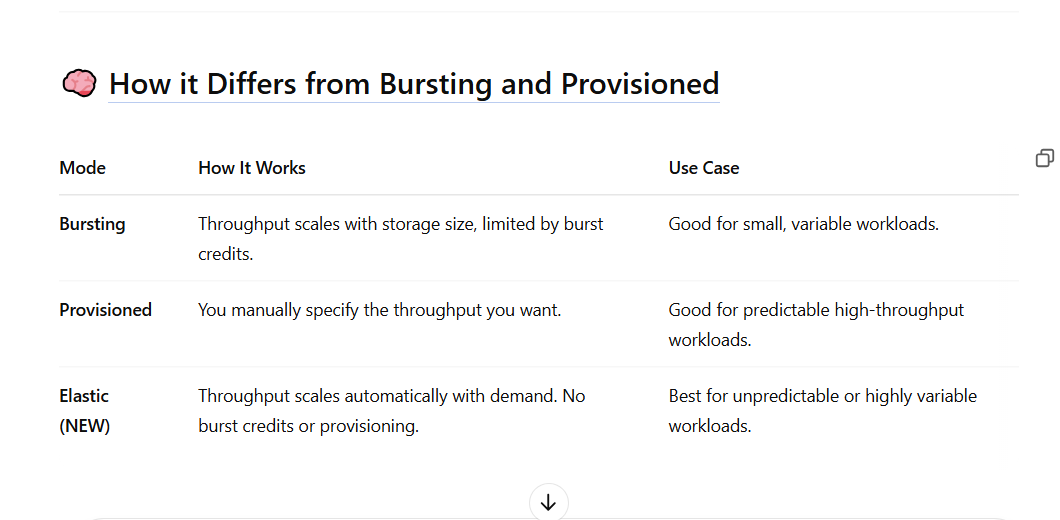
## 📌 ****Limits and Scaling****

* Automatically scales to petabytes.
* Supports thousands of concurrent clients via NFS v4.1 and v4.2.
* Supports **up to 25 GB/s** throughput and **over 1 million IOPS** (in Max I/O mode with large scale).

## 🧩 ****Use Cases Comparison****

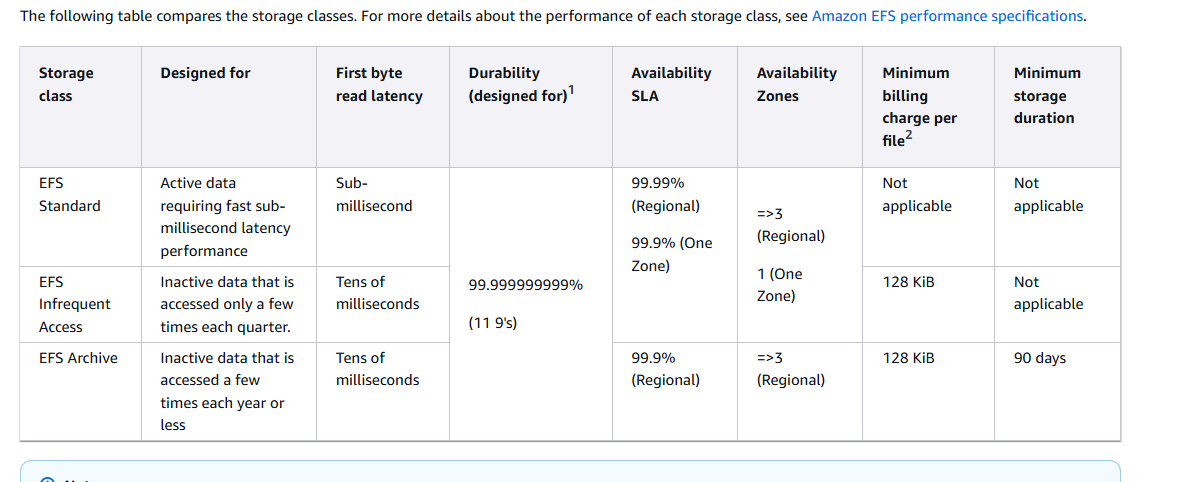
| **Workload Type** | **Suggested Performance Mode** | **Suggested Throughput Mode** |
| --- | --- | --- |
| Web servers | General Purpose | Bursting |
| Development environments | General Purpose | Bursting |
| Machine learning pipelines | Max I/O | Provisioned (if needed) |
| Data analytics jobs | Max I/O | Provisioned or Bursting |
| Video rendering / media | Max I/O | Provisioned |

How it Differs from Bursting and Provisioned



# 154 EFS Storage Class

#### storage classes



# 155 An instance profile

Great question! An **Instance Profile** in AWS is how you **attach an IAM role to an EC2 instance** so that the instance can access AWS resources securely.

### 🔹 So, what is an Instance Profile?

An **Instance Profile** is a **container for an IAM role** that can be attached to an EC2 instance.

Think of it like this:

rust

Sao chépChỉnh sửa

EC2 Instance <-- attached --> Instance Profile <-- contains --> IAM Role

### 🔸 Why do we need it?

* IAM roles are general-purpose (can be used by Lambda, ECS, etc.).
* **EC2 specifically needs an instance profile** as a wrapper around the IAM role.
* When you assign a role to an EC2 instance, under the hood, AWS attaches the **instance profile**, not just the role.

### 🧠 Key points:

* Each instance profile **can contain only one IAM role.**
* It allows the EC2 instance to get **temporary security credentials** from the **Instance Metadata Service (IMDS)**.
* These credentials allow the instance to call AWS services as per the role’s policy.

# 156 \*Placement groups feature

## Rules and limitations

Before you use placement groups, be aware of the following rules:

* An instance can be placed in one placement group at a time; you can't place an instance in multiple placement groups.
* You can't merge placement groups.
* [On-Demand Capacity Reservations](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-capacity-reservations.html#capacity-reservations-limits) and [zonal Reserved Instances](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/reserved-instances-scope.html) allow you to reserve capacity for EC2 instances in Availability Zones. When you launch an instance, if the instance attributes match those specified by an On-Demand Capacity Reservation or a zonal Reserved Instance, then the reserved capacity is automatically used by the instance. This is also true if you launch the instance into a placement group.
* You can't launch Dedicated Hosts in placement groups.
* You can't launch a Spot Instance that is configured to stop or hibernate on interruption in a placement group.

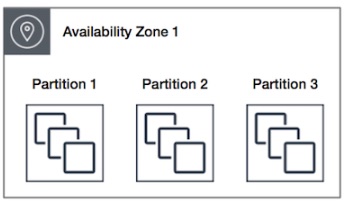
Top of Form

**Partition placement groups** - Partition placement groups help reduce the likelihood of correlated hardware failures for your application. When using partition placement groups, Amazon EC2 divides each group into logical segments called partitions. Amazon EC2 ensures that each partition within a placement group has its own set of racks. Each rack has its own network and power source. No two partitions within a placement group share the same racks, allowing you to isolate the impact of a hardware failure within your application.

The following image is a simple visual representation of a partition placement group in a single Availability Zone. It shows instances that are placed into a partition placement group with three partitions—Partition 1, Partition 2, and Partition 3. Each partition comprises multiple instances. The instances in a partition do not share racks with the instances in the other partitions, allowing you to contain the impact of a single hardware failure to only the associated partition.

Partition placement groups can be used to deploy large distributed and replicated workloads, such as HDFS, HBase, and Cassandra, across distinct racks. When you launch instances into a partition placement group, Amazon EC2 tries to distribute the instances evenly across the number of partitions that you specify. You can also launch instances into a specific partition to have more control over where the instances are placed.

A partition placement group can have partitions in multiple Availability Zones in the same Region. A partition placement group can have a maximum of seven partitions per Availability Zone. The number of instances that can be launched into a partition placement group is limited only by the limits of your account.

Partition placement groups:  via - <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html#placement-groups-partition>

Incorrect options:

**Cluster placement groups** - A cluster placement group is a logical grouping of instances within a single Availability Zone. A cluster placement group can span peered VPCs in the same Region. Instances in the same cluster placement group enjoy a higher per-flow throughput limit for TCP/IP traffic and are placed in the same high-bisection bandwidth segment of the network. Cluster placement groups are recommended for applications that benefit from low network latency, high network throughput, or both. They are also recommended when the majority of the network traffic is between the instances in the group. As the instances are packed close together inside an Availability Zone, this option is not correct for the given use case.

Cluster placement groups:  via - <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html#placement-groups-partition>

**Spread placement groups** - A spread placement group is a group of instances that are each placed on distinct racks, with each rack having its own network and power source. Spread placement groups are recommended for applications that have a small number of critical instances that should be kept separate from each other. Launching instances in a spread placement group reduces the risk of simultaneous failures that might occur when instances share the same racks. Spread placement groups provide access to distinct racks, and are therefore suitable for mixing instance types or launching instances over time. As the use-case talks about running large distributed and replicated workloads, so it needs more instances, therefore this option is not the right fit for the given use-case.

A spread placement group can span multiple Availability Zones in the same Region. You can have a maximum of seven running instances per Availability Zone per group.

The following image shows seven instances in a single Availability Zone that are placed into a spread placement group. The seven instances are placed on seven different racks.

Spread placement groups:  via - <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html#placement-groups-partition>

**Multi-AZ placement groups** - This is a made-up option, given as a distractor. You should note that the Partition and Spread placement groups can span across multiple Availability Zones in the same Region.

Reference:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html>

Bottom of Form

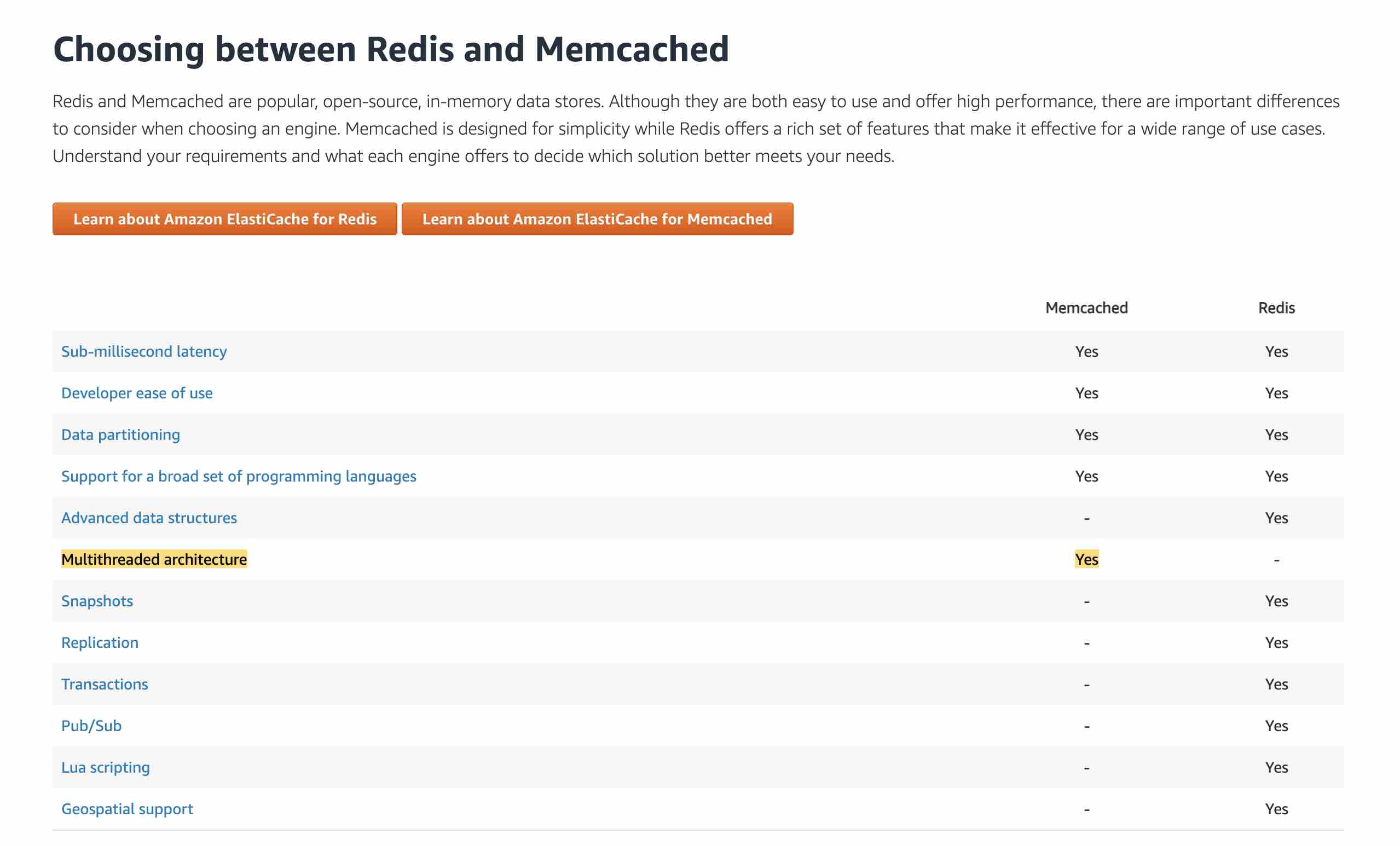
Top of Form

Bottom of Form

# 157 Why didn’t Amazon EC2 Auto Scaling stop an unhealthy instance?

<https://repost.aws/knowledge-center/auto-scaling-terminate-instance>

# 158 choise between Redis and memcahe



# 159 CloudFoundation Stack Template

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html>

## The Resources section is a required top-level section in a CloudFormation template. It declares the AWS resources that you want CloudFormation to provision and configure as part of your stack.

## Syntax

* The Resources section uses the following syntax:

### JSON

* "Resources" : {
* "*LogicalResourceName1*" : {
* "Type" : "AWS::*ServiceName::ResourceType*",
* "Properties" : {
* *"PropertyName1" : "PropertyValue1"*,
* *...*
* }
* },
* "*LogicalResourceName2*" : {
* "Type" : "AWS::*ServiceName::ResourceType*",
* "Properties" : {
* *"PropertyName1" : "PropertyValue1"*,
* *...*
* }
* }
* }

## Params

* Use the optional Parameters section to customize your templates. With parameters, you can input custom values to your template each time you create or update a stack. By using parameters in your templates, you can build reusable and flexible templates that can be tailored to specific scenarios.
* By defining parameters of the appropriate type, you can choose from a list of identifiers of existing resources when you use the console to create your stack. For more information, see [Specify existing resources at runtime with CloudFormation-supplied parameter types](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cloudformation-supplied-parameter-types.html).
* Parameters are a popular way to specify property values of stack resources. However, there may be settings that are region dependent or are somewhat complex for users to figure out because of other conditions or dependencies. In these cases, you might want to put some logic in the template itself so that users can specify simpler values (or none at all) to get the results that they want, such as by using a mapping. For more information, see [CloudFormation template Mappings syntax](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/mappings-section-structure.html).

## Syntax

* You declare parameters in a template's Parameters section, which uses the following general syntax:

### JSON

* "Parameters" : {
* "*ParameterLogicalID*" : {
* "*Description*": "*Information about the parameter*",
* "Type" : "*DataType*",
* "*Default*" : "*value*",
* "*AllowedValues*" : ["*value1*", "*value2*"]
* }
* }

# 160 AWS CloudHSM

## 🌩️ What is AWS CloudHSM?

**AWS CloudHSM (Hardware Security Module)** is a **cloud-based security appliance** that provides **secure key storage and cryptographic operations**. It's a **dedicated, single-tenant hardware appliance**, hosted by AWS, which you can use to meet **strict compliance and security requirements**.

Think of it like a locked, tamper-proof vault in AWS where you can store your encryption keys and perform secure operations—but only you have the keys to open it.

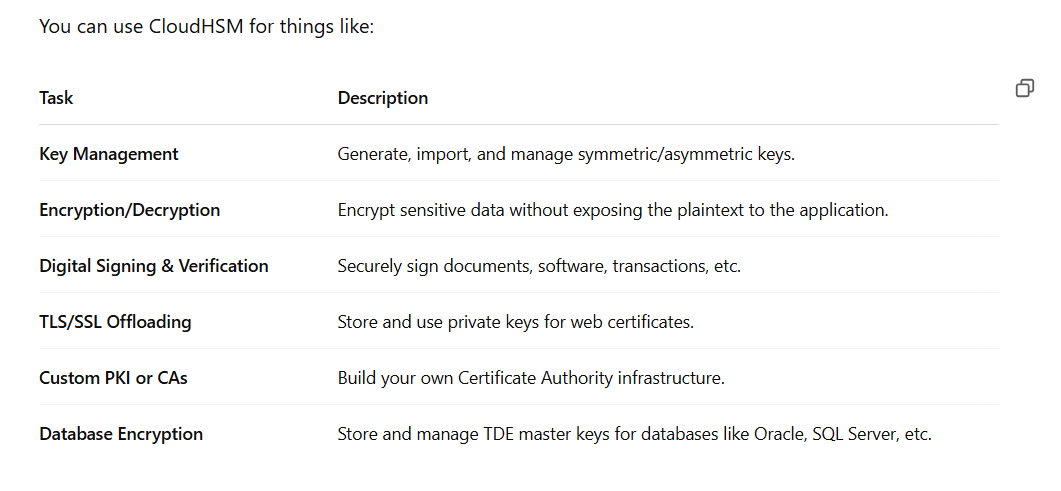
## 🔐 Why Use CloudHSM?

You might need CloudHSM if you:

* Need **full control of your cryptographic keys** (AWS **cannot** access them).
* Must meet **compliance standards** like **FIPS 140-2 Level 3**, **PCI DSS**, **HIPAA**, or **GDPR**.
* Want to **migrate on-prem HSM workloads** to the cloud.
* Use **custom or specialized cryptographic libraries** or algorithms not supported by AWS KMS.

## 🧰 What Can You Do With It?

You can use CloudHSM for things like:



## 🛠️ How It Works

### 1. **Provision HSM**

You create a CloudHSM **cluster** in a VPC. Each cluster includes **dedicated HSM appliances** in one or more Availability Zones.

### 2. **Initialize the Cluster**

* You become the **HSM owner** by creating an initial user called admin.
* No one else, including AWS, has access to your cluster or its keys.

### 3. **Manage Users & Roles**

You create **Crypto Users (CUs)** to perform key operations and **Admin Users (AUs)** to manage users and keys.

### 4. **Connect Your Applications**

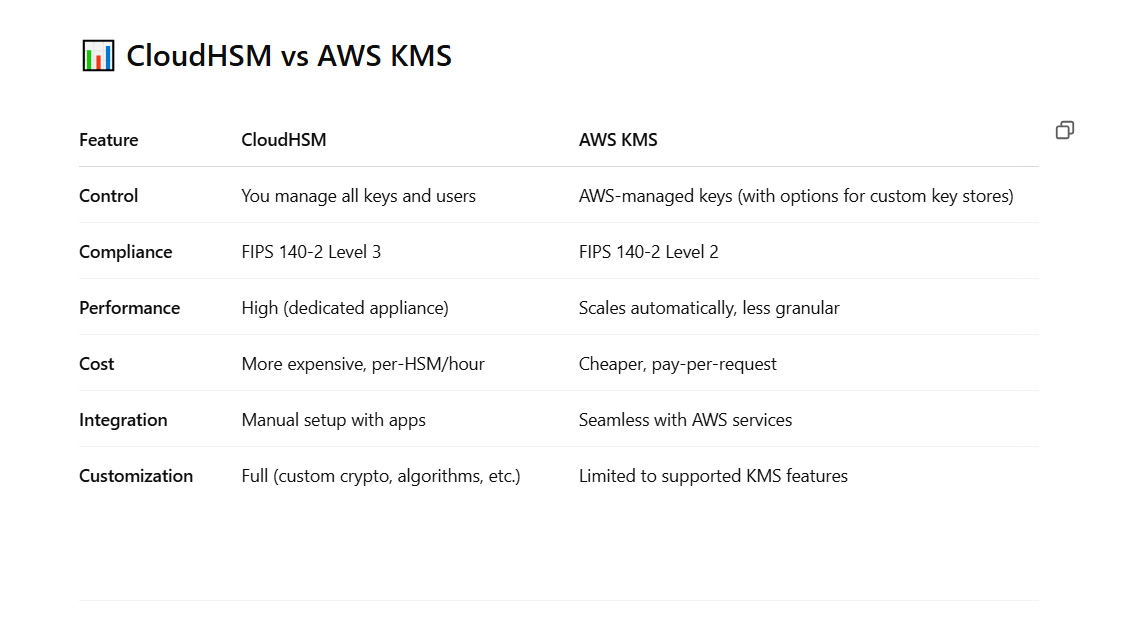
You install the **CloudHSM client** on your EC2 instance or application server, which communicates with the HSM over a secure channel.

### 5. **Use Crypto APIs**

Use libraries such as:

* **PKCS#11** – C-based standard API for cryptographic operations.
* **Java JCE** – Java Cryptography Extension.
* **Microsoft CNG** – Cryptographic Next Generation for .NET/Windows apps.

📊 CloudHSM vs AWS KMS



# 161 POSIX-compliant ( Trong NFS not for windown )

In the context of AWS, **POSIX-compliant** refers to storage or file systems that adhere to the **POSIX (Portable Operating System Interface)** standard. This standard defines a set of operating system-level interfaces, especially for **file and directory operations**, ensuring compatibility with Unix-like systems (e.g., Linux).

When something is **POSIX-compliant in AWS**, it means you can use it like a traditional file system — with standard file operations like:

* open(), read(), write(), close()
* Permissions and ownership (chmod, chown)
* Directory structures and symbolic links
* Locking, timestamps, atomic operations, etc.

# 162 Security Groups custome -default

+ There is an outbound rule that allows all traffic to all IP addresses

+ There are no inbound rules and traffic will be implicitly denied

# 163 NACL custome - default

+ There is a default inbound rule denying all traffic

+ There is a default outbound rule denying all traffic

* Network ACL’s function at the subnet level
* With NACLs you can have permit and deny rules
* Network ACLs have separate inbound and outbound rules and each rule can allow or deny traffic
* Network ACLs are stateless so responses are subject to the rules for the direction of traffic
* NACLs only apply to traffic that is ingress or egress to the subnet not to traffic within the subnet
* A VPC automatically comes with a default network ACL which allows all inbound/outbound traffic
* A custom NACL denies all traffic both inbound and outbound by default

# 164 Schedule your containers on Amazon ECS

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/scheduling_tasks.html>

* Amazon ECS provides a service scheduler (for long-running tasks and applications), the ability to run tasks manually (for batch jobs or single run tasks), with Amazon ECS placing tasks on your cluster for you
* The service scheduler is ideally suited for long running stateless services and applications
* Amazon ECS allows you to create your own schedulers that meet the needs of your business, or to leverage third party schedulers
* Blox is an open- source project that gives you more control over how your containerized applications run on Amazon ECS
* Blox enables you to build schedulers and integrate third-party schedulers with Amazon ECS while leveraging Amazon ECS to fully manage and scale your clusters
* Custom schedulers use the StartTask API operation to place tasks on specific container instances within your cluster
* Custom schedulers are only compatible with tasks using the EC2 launch type. If you are using the Fargate launch type for your tasks, the StartTask API does not work

## chatgpt

### Types of Scheduling in Amazon ECS

Amazon ECS provides **two main types** of scheduling strategies:

#### 1. **Service Scheduler**

This is the most commonly used scheduler.

* **Use Case**: Long-running services (like web servers or APIs)
* **Manages**:
  + Task placement
  + Auto-restarts on failure
  + Load balancing (with ALB or NLB)
  + Auto-scaling

plaintext

Sao chépChỉnh sửa

Example: "Keep 3 copies of my web app running at all times."

#### 2. **Task Scheduler (One-off tasks)**

Used for **short-lived or batch tasks**. You launch a task manually or via automation (like Step Functions, CloudWatch Events).

* No automatic restarts
* You choose where/how to run it
* Great for jobs like:
  + Data processing
  + Migrations
  + CRON jobs

plaintext

Sao chépChỉnh sửa

Example: "Run this image once to generate a report."

### ⚙️ Task Placement Strategies (for both schedulers)

You can control **how ECS places tasks** on your infrastructure:

* binpack: Packs tasks tightly based on CPU or memory (best for cost-saving)
* spread: Distributes tasks evenly (across AZs or instances)
* random: Places tasks randomly
* custom placement constraints: Use attributes (like instance type, environment, etc.)

# 165 ****EBS-Optimized = Dedicated Network for EBS****

* **EBS-Optimized instances** provide a **separate, dedicated network path** specifically for **EBS I/O traffic**.
* This avoids competition with regular instance network traffic (like internet, other AWS services).
* The goal is to ensure:
  + 💾 **Faster** and more **consistent EBS performance**
  + 🛡️ **Lower latency**
  + 📈 **Higher throughput and IOPS stability**

### 🔁 Think of it like:

🛣️ **Normal Instance:**  
EBS and app traffic share the same road — traffic jams happen.

🚗 **EBS-Optimized Instance:**  
EBS has its **own fast lane** — no interference.

### 🔧 Use With:

* **io1/io2 volumes** (Provisioned IOPS)
* **High-performance applications**
* **Databases, analytics, large-scale workloads**

So yes — **EBS-optimized = optimized network for EBS access.** 🧠💡

4o

# Một số note trong sách

## S3

- create bucket is private deafult

- . Bucket names can contain up to 63 lowercase letters, numbers, hyphens, and periods

- Objects can range in size from 0 bytes up to 5TB, and a single bucket can store an unlimited number of objects

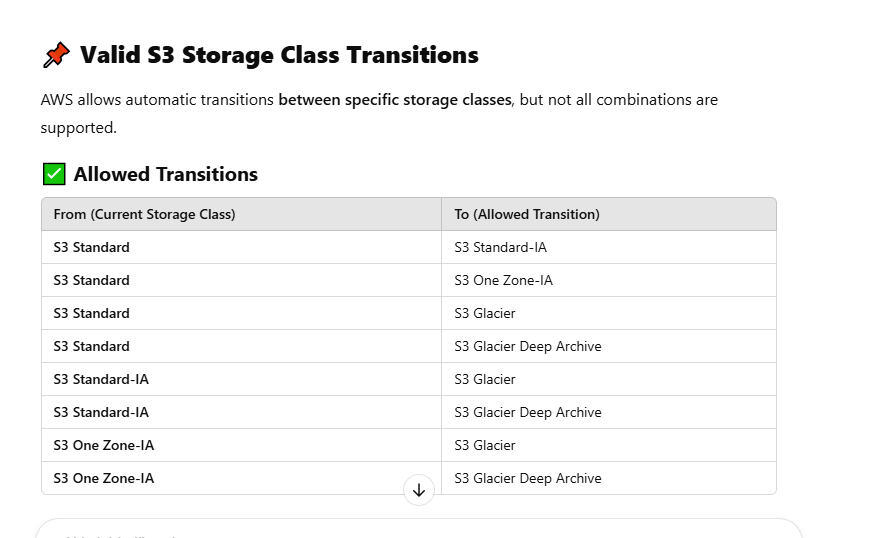
- Each object consists of data (the file itself) and metadata (data about the file). The data portion of an Amazon S3 object is opaque to Amazon S3. This means that an object’s data is treated as simply a stream of bytes—Amazon S3 doesn’t know or care what type of data you are storing, and the service doesn’t act differently for text data versus binary data. The metadata associated with an Amazon S3 object is a set of name/value pairs that describe the object.

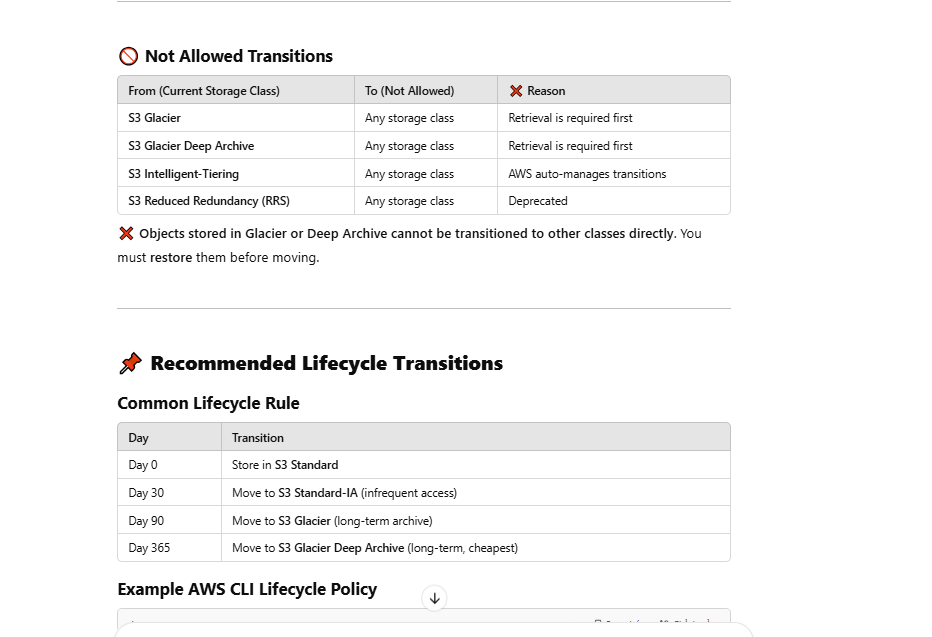
- Amazon S3 scales automatically, but for request rates over 100 GETS per second, it helps to make sure there is some randomness in the key space. Replication and logging will not affect performance or scalability. Using sequential key names could have a negative effect on performance or scalability.

- Amazon Glacier Is frequently erased within 90 days. Data is stored in encrypted archives that can be as large as 40TB. Archives typically contain TAR or ZIP files (a archives = 40TB), need 3-5 hours to restore

- Amazon Glacier can be used as a standalone service and as an Amazon S3 storage class.

### S3 Storage Class Transitions







## S3 **Amazon S3 storage classes**, focusing on **availability**, **durability**,

| **Storage Class** | **Durability** | **Availability** | **Retrieval Time** | **Min Storage Duration** | **Min Billable Object Size** | **Use Case** |
| --- | --- | --- | --- | --- | --- | --- |
| **S3 Standard** | 99.999999999% | 99.99% | Immediate | None | None | Frequently accessed data, general-purpose storage. |
| **S3 Intelligent-Tiering** | 99.999999999% | 99.9–99.99% | Immediate | 30 days | 128 KB (for auto-tiering) | Automatically moves data between tiers based on access. |
| **S3 Standard-IA** | 99.999999999% | 99.9% | Immediate | 30 days | 128 KB | Infrequently accessed data that needs fast access. |
| **S3 One Zone-IA** | 99.999999999% | 99.5% | Immediate | 30 days | 128 KB | Infrequent access in a single AZ, lower cost. |
| **S3 Glacier Instant Retrieval** | 99.999999999% | 99.9% | Milliseconds | 90 days | 128 KB | Archival with instant access. |
| **S3 Glacier Flexible Retrieval** | 99.999999999% | 99.99% | Minutes to hours | 90 days | 128 KB | Archive with flexible cost and access options. |
| **S3 Glacier Deep Archive** | 99.999999999% | 99.99% | Hours (12+) | 180 days | 128 KB | Lowest-cost storage for rarely accessed data. |
| **S3 Reduced Redundancy (Legacy)** | 99.99% | 99.99% | Immediate | None | None | Deprecated; use other classes. |

### Glacier

* Designed for durability of 99.999999999% of objects across multiple Availability Zones
* Data is resilient in the event of one entire Availability Zone destruction
* Supports SSL for data in transit and encryption of data at rest
* Extremely low cost design is ideal for long-term archive
* Lifecycle management for automatic migration of objects

## EC2

Max size of a volumn EBS is 16TB

#### Addressing an Instance:

- Public Domain Name System (DNS) Name—When you launch an instance, AWS creates a DNS name that can be used to access the instance. This DNS name is generated automatically and cannot be specified by the customer

- Public IP—A launched instance may also have a public IP address assigned. This IP address is assigned from the addresses reserved by AWS and cannot be specified

- Elastic IP—An elastic IP address is an address unique on the Internet that you reserve independently and associate with an Amazon EC2 instance

- Can modify type of instand :

Instance Type The ability to change the instance type of an instance contributes greatly to the agility of running workloads in the cloud

#### Security Groups:

default

+ There is an outbound rule that allows all traffic to all IP addresses

+ There are no inbound rules and traffic will be implicitly denied

If an instance is running in an Amazon VPC (discussed in Chapter 4), you can change which security groups are associated with an instance while the instance is running. For instances outside of an Amazon VPC (called EC2-Classic), the association of the security groups cannot be changed after launch.

* While termination instance but cli still can access instance. Turn on termination protection will disable termination in cli

. In order to prevent termination via the AWS Management Console, CLI, or API, termination protection can be enabled for an instance. While enabled, calls to terminate the instance will fail until termination protection is disabled. This helps to prevent accidental termination through human error.

#### Pricing Options

- On-Demand Instances

- Reserved Instances

+ All Upfront—Pay for the entire reservation up front. There is no monthly charge for the customer during the term.

+ Partial Upfront—Pay a portion of the reservation charge up front and the rest in monthly installments for the duration of the term.

+ No Upfront—Pay the entire reservation charge in monthly installments for the duration of the term.

- Spot Instances

#### Tenancy Options

- Shared Tenancy Shared tenancy is the default tenancy model for all Amazon EC2 instances, regardless of instance type, pricing model, and so forth

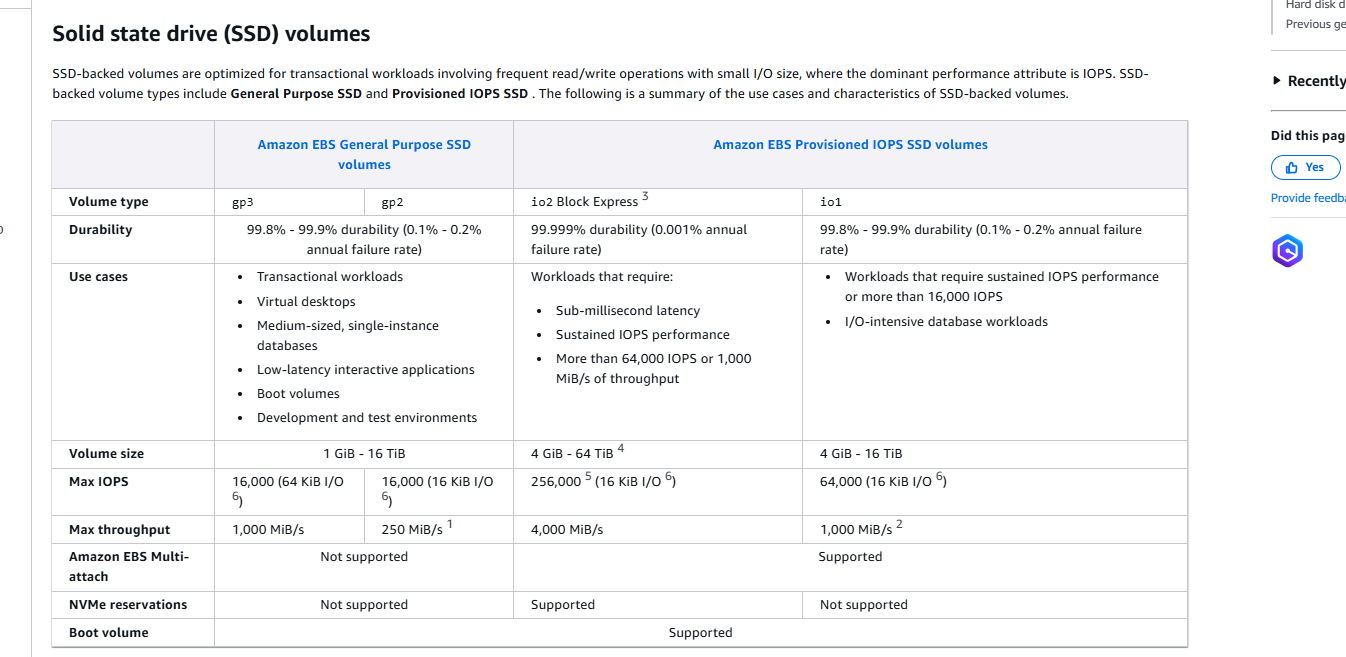
- Dedicated Instances Dedicated Instances run on hardware that’s dedicated to a single customer

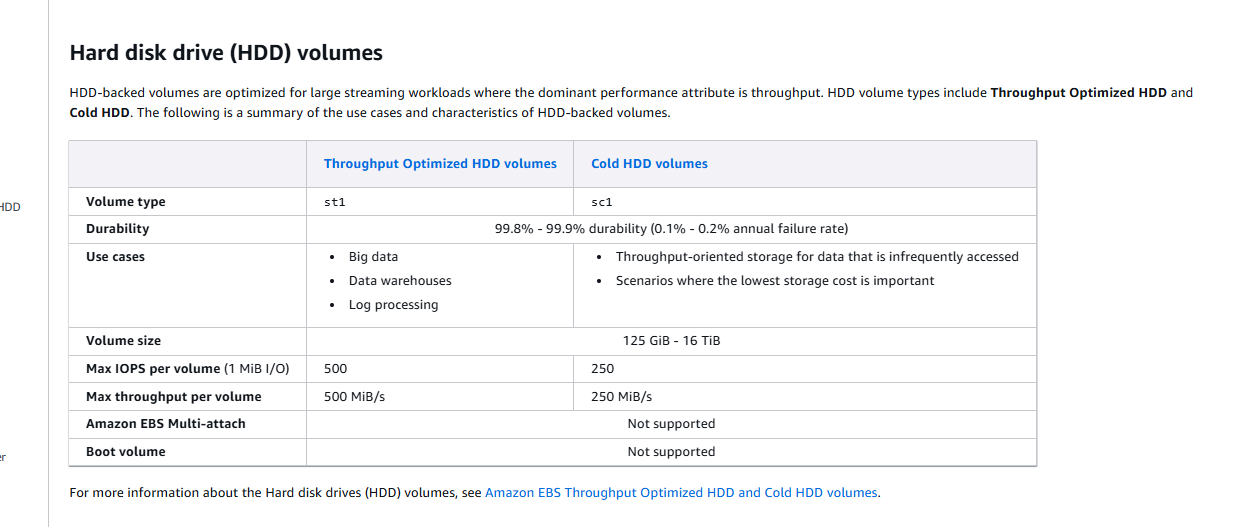
- Dedicated Host An Amazon EC2 Dedicated Host is a physical server with Amazon EC2 instance capacity fully dedicated to a single customer’s use.

#### Amazon Elastic Block Store (Amazon EBS)

- Magnetic Volumes

A magnetic Amazon EBS volume can range in size from 1 GB to 1 TB and will average 100 IOPS, but has the ability to burst to hundreds of IOPS. They are best suited for: Workloads where data is accessed infrequently Sequential reads Situations where low-cost storage is a requirement

* General-Purpose SSD 1GB- 16TB max 10000 IOPS, 1TB =3000IOPS ⬄ 3TB=10000IOPS
* Provisioned IOPS SSD Provisioned IOPS SSD volumes are designed to meet the needs of I/O-intensive workloads
* 



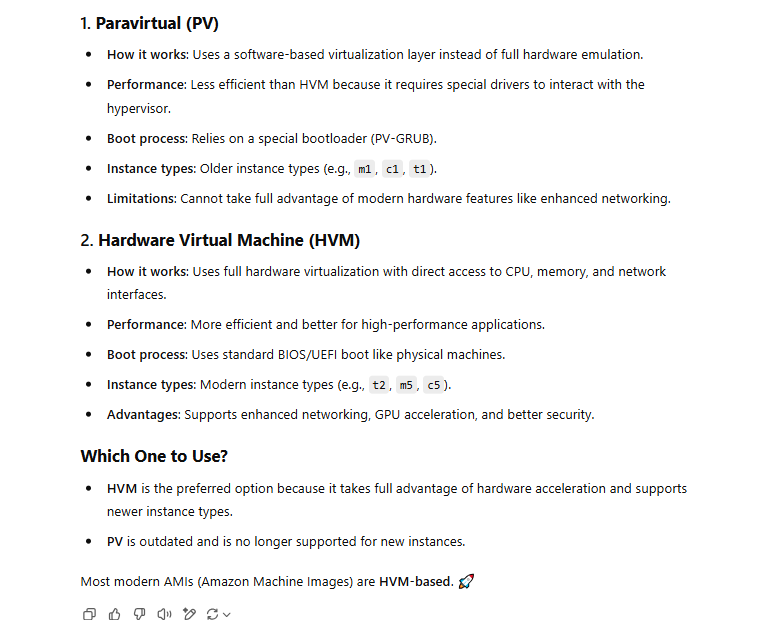
### Enhanced networking on Amazon EC2 instances

Link: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html>

Enhanced networking uses single root I/O virtualization (SR-IOV) to provide high-performance networking capabilities on supported instance types. SR-IOV is a method of device virtualization that provides higher I/O performance and lower CPU utilization when compared to traditional virtualized network interfaces. Enhanced networking provides higher bandwidth, higher packet per second (PPS) performance, and consistently lower latency between instances. There is no additional charge for using enhanced networking.

* Instances be launched from an HVM AMI (not PV)
* Is only supported in a VPC

### EC2 PV and HVM



### EC2 - Bastion Host

A Bastion Host is a secure instance that is used to access instances in a private subnet. The Bastion Host acts as a gateway to the private subnet and usually has minimal security exposure.

* 1 Computer được cấu hình đặc biệt, thuộc miền external/ public (DMZ) hoặc bên ngoài firewall, hoạt động như một server trung gian, cho phép bạn connect vào các Instance nằm trong Private Subnet
* Trường hợp Instance bị terminated, nhưng Auto Scaling Group đang launches, thì Elastic IP sẽ được đính lên cho instances mới

## VPC

### VPC Peering

### is used to enable private connectivity between VPCs within the same region or across different regions. It allows instances in one VPC to communicate with instances in another VPC using private IP addresses.

## EBL (ELATIC LOAD\_BALANLING)

### **Types of AWS Load Balancers**

AWS offers different types of load balancers based on the use case:

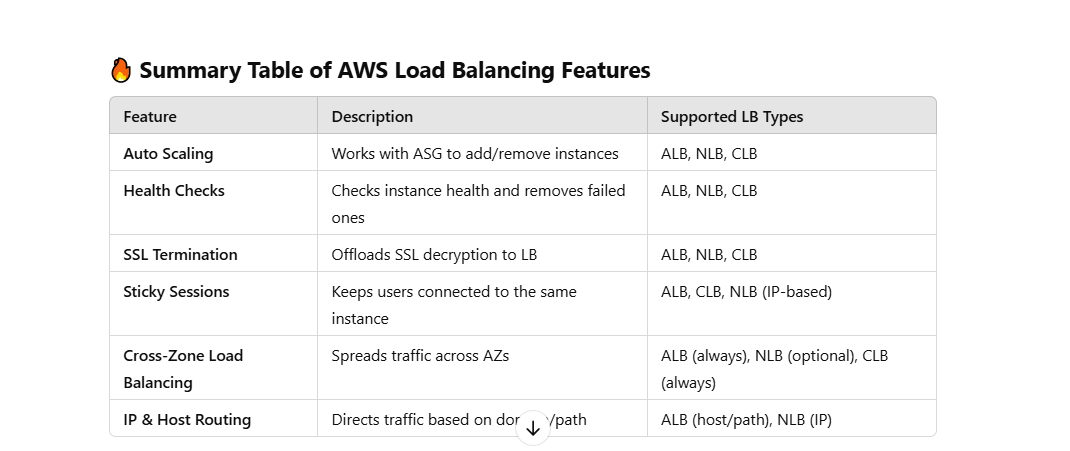
| **Load Balancer** | **Protocol** | **Best For** |
| --- | --- | --- |
| **Application Load Balancer (ALB)** | HTTP, HTTPS | Web applications (Layer 7 - Application layer) |
| **Network Load Balancer (NLB)** | TCP, UDP, TLS | High-performance, low-latency traffic (Layer 4 - Transport layer) |
| **Gateway Load Balancer (GWLB)** | IP | Third-party security and network appliances |
| **Classic Load Balancer (CLB)** (Legacy, not recommended) | HTTP, HTTPS, TCP | Basic load balancing |

### **2️⃣ How Load Balancers Work**

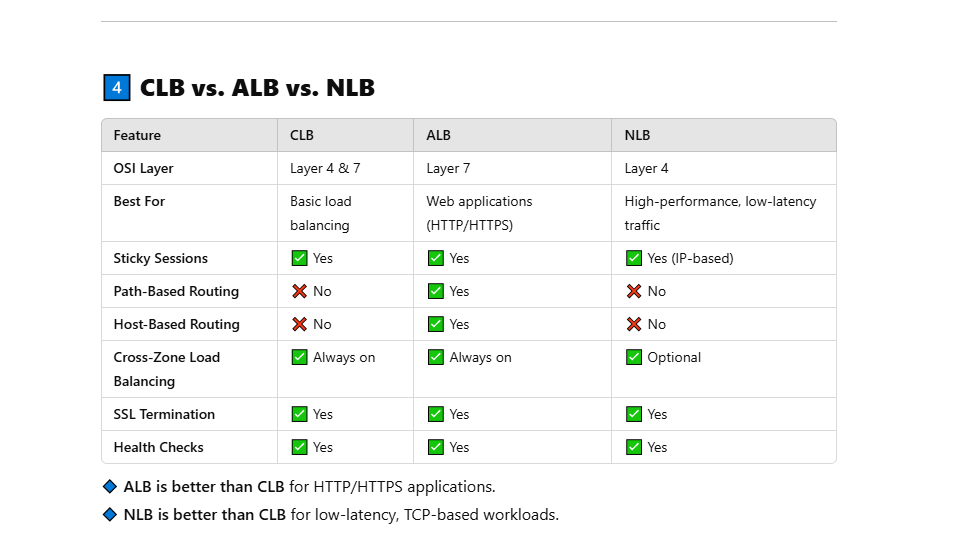
1. **Receives Requests**: A load balancer receives incoming traffic.
2. **Distributes Traffic**: It distributes requests across multiple backend instances.
3. **Performs Health Checks**: Ensures only healthy instances receive traffic.
4. **Handles Scaling**: Works with **Auto Scaling Groups** to dynamically add/remove instances.

### **3️⃣ Features of AWS Load Balancing**

✅ **Automatic Scaling**: Works with Auto Scaling Groups to handle traffic spikes.  
✅ **Health Checks**: Detects unhealthy instances and routes traffic only to healthy ones.  
✅ **SSL Termination**: Offloads SSL/TLS processing to the load balancer (ALB & NLB).  
✅ **Sticky Sessions**: Ensures a user session is always routed to the same instance.  
✅ **Cross-Zone Load Balancing**: Distributes traffic evenly across AZs.  
✅ **IP-based or Hostname-based Routing**: ALB allows routing based on host/path rules.



### Type of load balanling



## Public and private host Zone

### **When Do You Need** enableDnsHostnames **and** enableDnsSupport**?**

✅ Required for **Private Hosted Zones**, because AWS assigns **private DNS hostnames** for instances inside a **VPC**.  
❌ Not needed for **Public Hosted Zones**, since public records are already accessible over the internet.

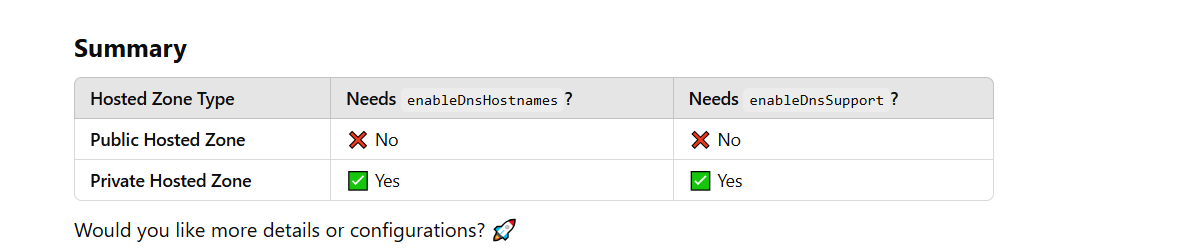
### **Example Scenarios**

#### ✅ **Using a Private Hosted Zone (**example.internal**)**

* enableDnsSupport = true
* enableDnsHostnames = true
* app.example.internal → Resolves to 10.0.1.50 (private EC2)

#### ❌ **Using a Public Hosted Zone (**example.com**)**

* **No need to enable enableDnsHostnames**
* www.example.com → Resolves to 203.0.113.10 (public EC2 or ALB)



#### **1.** enableDnsSupport **→ Allows Private DNS Resolution in VPC**

📌 **Purpose:**

* Ensures that the **AWS DNS resolver** (Amazon-provided .2 resolver) can resolve private domain names from Route 53.
* Without this setting, VPC instances cannot resolve **internal domain names** (e.g., app.example.internal → 10.0.1.100).

✅ **Example:**  
When enableDnsSupport = true, the EC2 instance **can resolve internal DNS names**:

sh

Sao chépChỉnh sửa

ping app.example.internal # Resolves to 10.0.1.100

If enableDnsSupport = false, the DNS lookup **fails** because Route 53 resolution is disabled.

#### **2.** enableDnsHostnames **→ Assigns DNS Hostnames to EC2 Instances**

📌 **Purpose:**

* Assigns **private DNS names** (e.g., ip-10-0-1-10.ec2.internal) to EC2 instances in the VPC.
* Required if you want to resolve **private DNS hostnames** inside the VPC.

✅ **Example:**  
When enableDnsHostnames = true, an EC2 instance in the VPC gets:

vbnet

Sao chépChỉnh sửa

Private IP: 10.0.1.10

Private DNS: ip-10-0-1-10.ec2.internal

This means **other EC2 instances** in the VPC can refer to it by hostname instead of IP.

If enableDnsHostnames = false, the EC2 instance **only has an IP** and no private hostname.

## RedShift (fully managed)

### **📌 Ad-Hoc Queries in Amazon Redshift**

🔹 **Ad-hoc queries** in Amazon Redshift are **one-time, exploratory SQL queries** that users run **on-demand** to analyze data **without predefined reports or dashboards**. These queries allow users to investigate trends, perform quick data checks, and gain insights without extensive ETL or pre-processing.

### **📌 Amazon Redshift Spectrum:**

#### **Query Data in S3 Without Loading It**

**Amazon Redshift Spectrum** is a feature that allows you to **query data directly from Amazon S3** **without loading it into Redshift**. It extends Redshift’s capabilities by allowing you to run **SQL queries on structured and semi-structured data** stored in S3, **without needing to store the data inside Redshift**.

🚀 **Main Benefit:** You can analyze vast amounts of data in S3 **without using Redshift storage**, reducing costs while maintaining performance.

#### **Key Features of Redshift Spectrum**

✅ **Query data in S3 without moving it into Redshift**  
✅ **Supports various file formats** (CSV, JSON, Parquet, ORC, Avro)  
✅ **Works with Athena and Glue** for metadata management  
✅ **Joins S3 data with Redshift tables**  
✅ **Pay only for the amount of data scanned**

#### **🚀 When to Use Redshift Spectrum**

✅ You have **large datasets** in S3 that you don't want to move into Redshift.  
✅ You need to **analyze infrequent or historical data** without increasing Redshift storage costs.  
✅ You want to **combine S3 and Redshift data** in a single SQL query.  
✅ You prefer **serverless data analysis** rather than managing a large Redshift cluster.

### **Redshift Data Sharing (Share Data Across Clusters)**

✅ **Share data across multiple Redshift clusters** **without duplication**.  
✅ Allows **real-time analytics** across departments, teams, or businesses.  
✅ Supports **cross-account data sharing** in AWS Organizations.

### **Amazon Redshift Serverless (Auto-Scaling, No Cluster Management)**

✅ **Fully managed serverless Redshift** → No need to provision clusters.  
✅ **Automatically scales compute resources** based on query demand.  
✅ **Pay only for the compute used** (no idle cluster costs).  
✅ Ideal for **event-driven analytics and unpredictable workloads**.

## IAM

An IAM user with full administrator access can perform almost all AWS tasks except a few tasks designated only for the root account user.

Some of the AWS tasks that only a root account user can do are as follows :

+ change account name or root password or root email address

+ change AWS support plan

+ close AWS account

+ enable MFA on S3 bucket delete

+ create Cloudfront key pair

+ register for GovCloud.

### Type Restrict access BY IAM

#### **1. Service Control Policies (SCPs)**

🔹 **Scope**: AWS Organizations  
🔹 **Purpose**: Restricts permissions at the **account or organizational unit (OU) level**.  
🔹 **Key Feature**:

* Even if an IAM user or role has a policy granting permissions, an **SCP can restrict them**.
* It does **not grant permissions**, only limits them.
* Used to enforce compliance across multiple AWS accounts.

✅ **Example**:

* An SCP allows only s3:ListBucket, even if an account administrator grants full S3 access to a user.

#### **2. IAM Policies**

🔹 **Scope**: IAM Users, Roles, and Groups  
🔹 **Purpose**: Grants or denies permissions explicitly to users, roles, or groups.  
🔹 **Key Feature**:

* Policies define what actions are **allowed or denied**.
* Can be **AWS Managed Policies** (predefined) or **Customer Managed Policies** (custom).

✅ **Example**:

* A policy allowing s3:PutObject but denying s3:DeleteObject.

#### **3. Resource-Based Policies**

🔹 **Scope**: AWS Resources (e.g., S3, SNS, SQS, Lambda)  
🔹 **Purpose**: Controls who can access a specific resource.  
🔹 **Key Feature**:

* Unlike IAM policies, which attach to users/roles, resource-based policies attach to AWS resources.
* Supports **cross-account access**.

✅ **Example**:

* An **S3 bucket policy** allowing another AWS account to read objects.

#### **4. Access Control Lists (ACLs)**

🔹 **Scope**: S3 Buckets, VPCs, and AWS Network Resources  
🔹 **Purpose**: Grants access at the **object level** rather than user/role level.  
🔹 **Key Feature**:

* Older and less flexible than IAM policies.
* Used for legacy access control.

✅ **Example**:

* An S3 ACL granting read-only access to a specific AWS account.

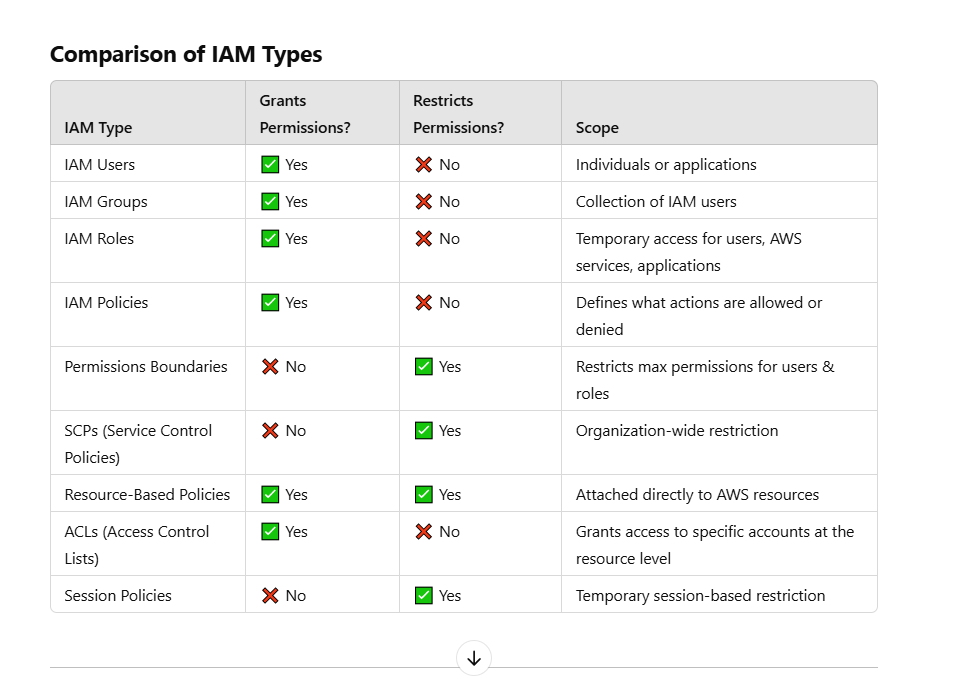
#### **5. Session Policies**

🔹 **Scope**: Temporary Security Credentials (via AssumeRole, STS, etc.)  
🔹 **Purpose**: Restricts permissions when a role is assumed.  
🔹 **Key Feature**:

* Applies additional restrictions on top of the role's permissions.
* Useful for granting **temporary, limited permissions**.

✅ **Example**:

* A session policy restricting a temporary IAM role to only s3:ListBucket

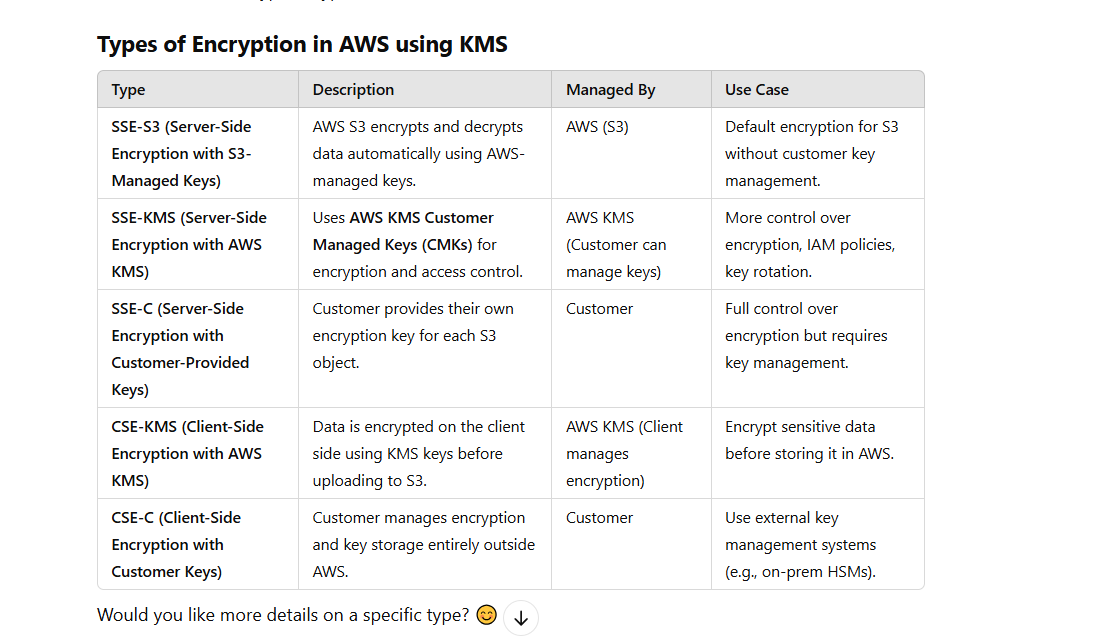


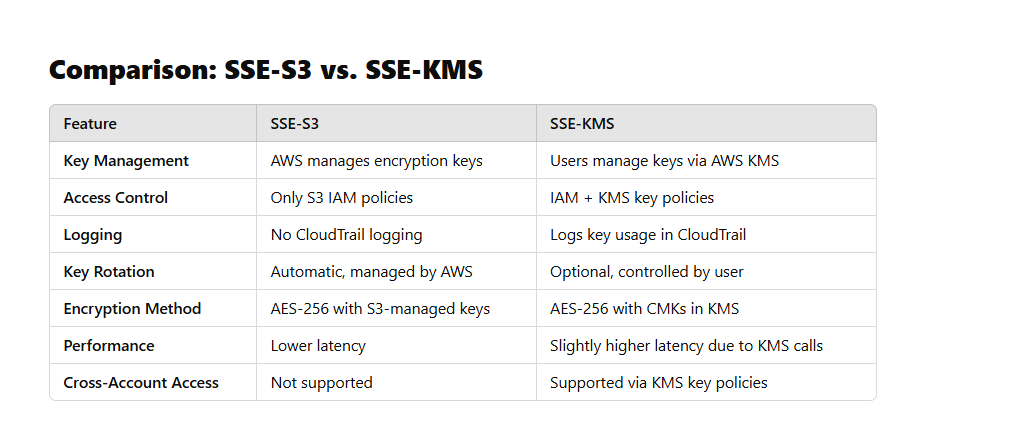
## IAM roles provide **secure, temporary** access to AWS resources for:

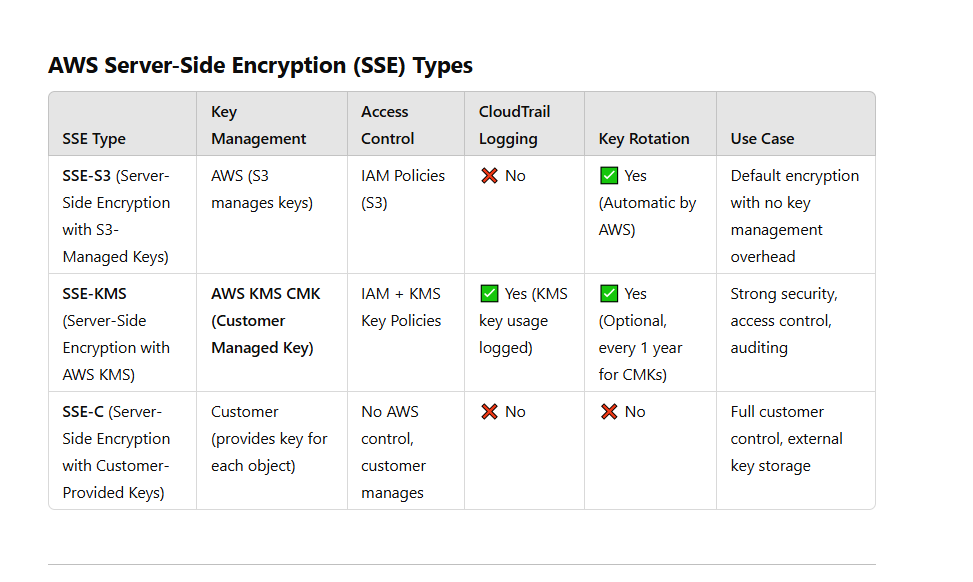


## SSE

* AWS Key Management Service (AWS KMS) is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud. When you use server-side encryption with AWS KMS (SSE-KMS), you can specify a customer managed CMK that you have already created. SSE-KMS provides you with an audit trail that shows when your CMK was used and by whom. Therefore SSE-KMS is the correct solution for this use-case.
* When you use Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3), each object is encrypted with a unique key. However this option does not provide the ability to audit trail the usage of the encryption keys.
* With Server-Side Encryption with Customer-Provided Keys (SSE-C), you manage the encryption keys and Amazon S3 manages the encryption, as it writes to disks, and decryption, when you access your objects.
* Using client side encryption is ruled out as the startup does not want to provide the encryption keys.







## Lamda

A runtime is a version of a programming language or framework that you can use to write Lambda functions. AWS Lambda supports runtimes for the following languages:

* C#/PowerShell
* Go
* Java
* Node.js
* Python
* Ruby

## Cloud front

### Restrict access from user using cloudfront

Amazon CloudFront provides multiple features to **restrict access** and control who can view your content. Here are the main methods:

#### **1. Signed URLs and Signed Cookies (Token-Based Access)**

* You can generate **signed URLs** or **signed cookies** to allow only authorized users to access content.
* These include an **expiration time** and **policy restrictions**.
* Ideal for paywalled content, private videos, and premium downloads.
* Example: Allow access to specific users for a limited time.

#### **2. Origin Access Control (OAC) / Origin Access Identity (OAI) for S3**

* Prevents direct access to an Amazon S3 bucket.
* Uses a special identity (OAI/OAC) to allow CloudFront to serve content securely.
* Users must access files **only via CloudFront**, not directly from S3.

#### **3. Geo-Restriction (Geoblocking)**

* Restricts access based on the user's country.
* Two modes:  
  ✅ **Whitelist:** Allow specific countries.  
  ❌ **Blacklist:** Block specific countries.
* Useful for region-specific content licensing.

#### **4. AWS WAF (Web Application Firewall) & Security Rules**

* Blocks or allows traffic based on IP addresses, headers, or custom rules.
* Protects against **DDoS attacks**, SQL injection, and XSS.

#### **5. Restrict Access with IAM Policies**

* If CloudFront is serving from an **AWS service (like API Gateway or S3)**, IAM policies can enforce restrictions.

#### **6. Restrict by HTTP Referer Header**

* Block access if the request does not come from your website.
* Helps prevent **hotlinking** (others embedding your content)

# II.Questions

## 2.1. Elastic Load Balancing supports which of the following types of load balancers? (Choose 3 answers)

* Internet-facing
* Internal
* Hypertext Transfer Protocol Secure (HTTPS) using Secure Sockets Layer (SSL)

##### **What is a network ACL in Amazon VPC?**

##### A network ACL is a firewall that controls traffic to and from subnets in a VPC. network ACL is a firewall that controls tr

## How can you speed up the propagation of a DNS change in Amazon Route 53?

##### affic to and Reduce the TTL value of the DNS record.

##### rom subnets in a VPC.

The TTL (Time to Live) value determines how long a DNS record can be cached by resolvers. By reducing the TTL value, you can speed up the propagation of a DNS change. However, decreasing the TTL value also increases the load on the DNS servers and increases the risk of DNS amplification attacks.

## What is the purpose of an alias record in Amazon Route 53?

To map a DNS name to an AWS resource, such as an ELB load balancer or an S3 bucket